UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA – DEE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEEL

ALEXANDRE ERWIN ITTNER

IMPLEMENTAÇÃO E AVALIAÇÃO DE ABORDAGENS HEURÍSTICAS PARA O PROBLEMA DO ROTEAMENTO DE CABOS EM PAINÉIS ELÉTRICOS

Joinville
2010

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA – DEE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEEL

ALEXANDRE ERWIN ITTNER

IMPLEMENTAÇÃO E AVALIAÇÃO DE ABORDAGENS HEURÍSTICAS PARA O PROBLEMA DO ROTEAMENTO DE CABOS EM PAINÉIS ELÉTRICOS

Dissertação submetida à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Engenharia Elétrica

Orientador: Dr. Roberto Silvio Ubertino Rosso Jr.
Co-Orientador: Dr. Claudio Cesar de Sá

Joinville
2010

ALEXANDRE ERWIN ITTNER

IMPLEMENTAÇÃO E AVALIAÇÃO DE ABORDAGENS HEURÍSTICAS PARA O PROBLEMA DO ROTEAMENTO DE CABOS EM PAINÉIS ELÉTRICOS

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovado em sua forma final pelo Curso de Mestrado Profissional em Engenharia Elétrica do CCT/UDESC.

Banca Examinadora:

Orientador:

_____

Dr., Roberto Silvio Ubertino Rosso Jr.

Membro:

_____

Dr., Ademir Nied

Membro:

_____

Dra., Jerusa Marchi

Joinville, 24 de Agosto de 2010

I91i

Ittner, Alexandre Erwin

Implementação e Avaliação de Abordagens Heurísticas para o Problema do Roteamento de Cabos em Painéis Elétricos / Alexandre Erwin Ittner; Orientador: Roberto Silvio Ubertino Rosso Jr. – Joinville, 2010

123 f. : il ; 30 cm.
Incluem referências.

Dissertação (mestrado) – Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Mestrado Profissional em Engenharia Elétrica, Joinville, 2010

1. Otimização 2. Quadros elétricos 3. Condutores elétricos 4. Automação industrial 5. Inteligência artificial 6. Algoritmos genéticos I. Rosso, Roberto S. U., Jr. II. Título

CDD 629.8

Aos meus pais e à minha "oma"

AGRADECIMENTOS

À minha família e amigos pelo apoio integral e incentivo;

Aos meus companheiros da Valinor: Dáfnie, Débora, Eduardo, Felipe, Fábio, Fabio, Flávia, Gabriel, Gilson, Henrique, Juliano, Reinaldo e Stefanie, a quem acostumei-me a chamar respectiva e carinhosamente de *Derinthuck Storm Gilgamesh, Valië Nion, Cildraemoth, Maider Stharlyon, Noel Anor, Deriel, Daiana, Tilion, Eleuvis Storm Gilgamesh, Alatar Anor, Setzer, Imrahil* e *Liudriska Storm*.

Aos professores Claudio Cesar de Sá, Prof. Roberto Silvio Ubertino Rosso Jr. e Fernando Deeke Sasse pelas orientações, direcionamentos e apoio.

Aos professores Antonio Heronaldo de Sousa, André B. Leal, Marcelo da Silva Hounsell e Sérgio Haffner pela parceria durante o curso.

A Sigmundo Preissler Junior, Yuri Kaszubowski Lopes, Gilsiley Henrique Darú, Flavio Bernardes, Valter Luiz Knihs, Mike Pall, Roberto Ierusalimschy, Igor Kondrasovas, Ruben Viegas e Roman Barták pelas informações trocadas durante o desenvolvimento deste trabalho.

A Kazumi Nakamatsu, Germano Lambert Torres, Jair Minoro Abe, Guilherme de Alencar Barreto e aos revisores anônimos pelas críticas construtivas e sugestões oferecidas durante a avaliação das publicações geradas nesta pesquisa.

Yé! Utúvienyes!

– Aragorn Telcontar

RESUMO

ITTNER, Alexandre Erwin. **Implementação e Avaliação de Abordagens Heurísticas para o Problema do Roteamento de Cabos em Painéis Elétricos.** 2010. 123f. Dissertação (Mestrado Profissional em Engenharia Elétrica – Área: Automação da Manufatura) – Universidade do Estado de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2010.

Esta dissertação apresenta um estudo sobre as características do Problema do Roteamento de Cabos em Painéis Elétricos e sua solução por meios computacionais. Especificamente, este trabalho apresenta uma definição formal para o problema, descreve as abordagens computacionais disponíveis e propõe uma série de algoritmos para sua solução. Por fim, descreve-se um aplicativo desenvolvido empregando os algoritmos propostos que permite a obtenção de bons resultados para as instâncias deste problema tipicamente encontradas na indústria.

Palavras-chave: Inteligência Artificial, Roteamento, Algoritmos Genéticos, Otimização por Colônias de Formigas.

ABSTRACT

ITTNER, Alexandre Erwin. **Implementation and Evaluation of Heuristic Approaches for the Cable Routing Problem in Electrical Panels.** 2010. 123p. Dissertation (Master in Electrical Engineering – Area: Manufacturing Automation) – Santa Catarina State University, Post Graduation Program in Electrical Engineering, Joinville (Brazil), 2010.

This dissertation presents a research work on the Cable Routing Problem in Electrical Panels and its resolution by computational means. Strictly, this work shows a formal definition for the problem, elaborates on the available computational approaches, and suggests several algorithms for its resolution. At last, an application developed using the proposed algorithms is described, yielding good results for the problem instances typically found in the industry.

Keywords: Artificial Intelligence, Routing, Genetic Algorithms, Ant Colony Optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS

Termo	Significado
ACO	*Ant Colony Optimization*
AG	Algoritmos Genéticos
AS	*Ant System*
BGA	*Breeder Genetic Algorithm*
CAD	*Computer Aided Design*
CAE	*Computer Aided Engineering*
CLP	*Constraint Logic Programming*
CP	*Constraint Programming*
CX	*Cycle Crossover*
DBGA	*Distributed Breeder Genetic Algorithm*
ER	*Genetic Edge Recombination Crossover*
Ex	Busca Exaustiva
FF	*First Fail*
HS	Heurística Simples
IA	Inteligência Artificial
IAG	Inserção por Algoritmos Genéticos
IFF	Inserção com *First Fail*
KP	*Knapsack Problem*
$\mathcal{MMAS}$	$\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*
MPX	*Maximal Preservative Crossover*
OX1	*Order Crossover*
OX2	*Order Based Crossover*
PCV	Problema do Caixeiro Viajante
PELV	*Protected extra-low voltage*
PM	Problema da Mochila
PL	Programação em Lógica
PLR	Programação em Lógica por Restrições
PMX	*Partially Mapped Crossover*

Termo	**Significado**
PO	Pesquisa Operacional
POS	*Position Based Crossover*
PRCPE	Problema do Roteamento de Cabos em Painéis Elétricos
PRV	Problema de Roteamento de Veículos
SM	*Swap-mutate*
TSP	*Travelling Salesman Problem*
VRP	*Vehicle Routing Problem*

LISTA DE SÍMBOLOS

Símbolo	Significado
$(\ldots)$	Tupla
$[\ldots]$	Lista
$A = \{a_1, a_2, \ldots, a_n\}$	Conjunto de arestas
A	Ponto inicial de um conduíte
$\overline{AB}$	Segmento de reta entre os pontos A e B
avg	Grau médio dos nós de um grafo
B	Ponto final de um conduíte
c	Custo por unidade de comprimento do cabo
c_t	Coeficiente de truncamento
c_{total}	Custo total
c_{unit}	Custo unitário
$compr(r_m)$	Comprimento m-ésima rota
$compr(x)$	Função de comprimento
$custo(S)$	Função de custo de uma solução
d_{ij}	Distância entre cidades (PCV) ou comprimento da aresta
$dist(x,y)$	Função de distância euclidiana
e_i	Dados elétricos de um cabo
$f(s)$	Função custo de uma solução
$f(s_{opt})$	Função custo de uma solução ótima
$G = (N, A)$	Grafo de construção
I_n	n-ésima interligação
L_c	Lista de conduítes
L_i	Lista de interligações
L_k	Comprimento total (custo) de uma solução
L_n	Lista de nós
L_t	Lista de terminais de ligação
L_w	Lista de cabos
M	Número real arbitrariamente grande

Símbolo	**Significado**
$N = \{n_1, n_2, \ldots, n_n\}$	Conjunto de componentes
n	Número de cidades (PCV)
n_c	Identificação do componente
n_{final}	Nó final
$n_{inicial}$	Nó incial
n_n	n-ésimo nó
$\mathcal{NP}$	Tempo polinomial não-determinístico
n_t	Identificação do terminal
$O(n)$	Ordem de complexidade de um algoritmo
$\mathcal{P}$	Tempo polinomial
p_{best}	Probabilidade de escolha da melhor aresta disponível
p_{ij}	Probabilidade de escolha de uma cidade de destino
p_m	Probabilidade de mutação
P_n	n-ésimo painel
P_{xyz}	Coordenadas espaciais de um ponto P
Q	Parâmetro configurável (ACO)
q_{cabos}	Número de cabos
q_{term}	Número de terminais
r	Valor aleatório
r_n	n-ésima rota
$R(w_k)$	Função associativa da rota de um cabo
s_c	Área da seção transversal interna de um conduíte
s_e	Área da seção transversal externa à isolação do cabo
s_{opt}	Solução ótima
$sect(w)$	Função de área da seção transversal externa de um cabo
t	Instante de tempo
t_c	Tipo do conduíte
t_d	Terminal de destino
t_f	Terminal final
t_i	Terminal inicial
T_n	n-ésimo terminal
t_o	Terminal de origem

Símbolo	**Significado**
w_n	n-ésimo cabo
α	Importância relativa (peso) da quantidade de feromônio
β	Importância relativa (peso) da função heurística
δ	Estado do problema em um grafo de construção
η	Atratividade de uma solução
η_{ij}	"Visibilidade" da cidade no PCV, tal que $\eta_{ij} = 1/d_{ij}$
ρ	Persistência da trilha de feromônio, tal que $\rho = 1 - \tau_{ev}$
τ	Quantidade de feromônio
τ_{ev}	Coeficiente de evaporação do feromônio
τ_{ij}	Intensidade da trilha de feromônio na aresta ij
τ_{incr}	Incremento da trilha de feromônio
τ_{max}	Intensidade máxima da trilha de feromônio
τ_{min}	Intensidade mínima da trilha de feromônio

SUMÁRIO

1 Introdução

Em grandes plantas industriais, como usinas hidrelétricas, montadoras automotivas e demais instalações com sistemas de automação industrial de médio e grande porte, existem painéis e quadros elétricos que empregam dezenas ou centenas de componentes interligados por vários milhares de cabos alojados em centenas de conduítes distribuídos na forma de um grafo. A distribuição desses cabos influencia o custo da instalação e organização estética dos painéis, impactando sobre a eficiência e a qualidade final do projeto. A definição adequada das rotas percorridas pelos cabos também permite estimar, em tempo de projeto, a quantidade de cada tipo de cabo usado nos painéis, permitindo a racionalização da compra do material.

O roteamento pode ser feito de forma artesanal: o projetista determina o menor caminho para cada cabo de forma empírica ou com auxílio de recursos de medida disponíveis (régua sobre um desenho em escala ou ferramentas de medida em uma aplicação CAD), começando pelos cabos mais caros e, progressivamente, para os mais baratos. Se não houver espaço disponível nos conduítes necessários para a rota mais curta entre dois componentes, o projetista desvia os cabos pela melhor rota alternativa disponível. Caso inexistam rotas viáveis, o projetista remaneja os cabos já roteados por outro caminho. O processo segue iterativamente até que todos os cabos sejam roteados.

O processo manual previamente descrito é desgastante, repetitivo e sujeito a erros humanos, com resultados aquém das necessidades exigidas pelas técnicas contemporâneas de engenharia. Este trabalho propõe um estudo sobre as propriedades envolvidas na disposição dos cabos em painéis que resultará em métodos computacionais para a otimização de rotas possibilitando a redução do custo global dos cabos do sistema.

1.1 Contribuições

Este trabalho elabora um estudo sobre as características do Problema do Roteamento de Cabos em Painéis Elétricos e sua solução por meios computacionais. Especificamente, este trabalho contribui com (1) a criação de uma definição formal para o Problema do Roteamento de Cabos em Painéis Elétricos, (2) a elaboração de uma estratégia para tratar eficientemente

a saturação parcial de conduítes em problemas de roteamento de cabos, (3) a definição de algoritmos heurísticos eficientes para a solução deste problema e (4) o desenvolvimento de um aplicativo que, empregando os algoritmos desenvolvidos, permite a obtenção de bons resultados para as instâncias deste problema tipicamente encontradas na indústria em um tempo de execução aceitável para os parâmetros do *software*.

1.2 Organização do Trabalho

Este trabalho é dividido conforme a estrutura a seguir: no Capítulo 2, faz-se uma revisão da literatura pertinente ao tema em visa à apresentação dos conceitos fundamentais necessários para a construção da definição formal do Problema do Roteamento de Cabos em Painéis Elétricos apresentada no Capítulo 3 e das técnicas descritas nos capítulos subsequentes. O Capítulo 4 descreve os algoritmos desenvolvidos para a solução do problema e a sua implementação em uma plataforma computacional. Testes e avaliações dos resultados obtidos com estes algoritmos são descritos no Capítulo 5 e, no Capítulo 6, expõem-se as conclusões e considerações finais.

2 Revisão da Literatura

Este capítulo apresenta os conceitos e técnicas necessários para a formalização do Problema do Roteamento de Cabos em Painéis Elétricos. Uma revisão da literatura relacionada aos problemas clássicos aplicados na sua formulação é feita nas Seções 2.1, 2.2 e 2.3 e as técnicas usadas para sua solução são examinadas nas Seções 2.4 e 2.5.

2.1 Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV, *Traveling Salesman Problem*, *TSP*) é um dos problemas mais estudados da área de otimização combinatória. Uma descrição sumária do problema, usando a terminologia da Teoria de Grafos, é: dado um grafo valorado, completo e não dirigido, encontrar um caminho que parta de um nó, visite todos os outros nós exatamente uma vez e retorne à origem, tal que a soma dos pesos das arestas usadas neste caminho é mínimo – ou seja, encontrar um ciclo hamiltoniano de custo mínimo. O nome do problema deriva da descrição informal onde um caixeiro viajante deve visitar um determinado número de cidades e voltar à cidade de origem percorrendo a menor distância possível (CORMEN et al., 2001).

O PCV é um problema da classe de complexidade $\mathcal{NP}$-completo, logo, supondo $\mathcal{P} \neq \mathcal{NP}$, o problema não possui uma solução computacionalmente eficiente (PAPADIMITRIOU; STEIGLITZ, 1998). Como o tempo de execução de um algoritmo para solução do PCV cresce exponencialmente em função do número de nós do grafo, a aplicação de buscas exaustivas torna-se inviável mesmo em instâncias com poucas dezenas de nós (PAPADIMITRIOU; STEIGLITZ, 1998). A importância deste problema deriva da frequência com que ele é encontrado em aplicações práticas, tanto na sua formulação original (casos clássicos como a geração da sequência de perfuração de placas de circuito impresso e cristalografia por raios-X) quanto em problemas derivados.

Há uma extensa literatura de referência disponível para o Problema do Caixeiro Viajante, como (APPLEGATE et al., 2007), e este problema é frequentemente usado para avaliação de algoritmos de otimização combinatória. Para fins de comparação, estas avaliações normalmente usam instâncias padronizadas, como as fornecidas pela TSPLIB (REINELT, 2009).

Abordagens para o problema do caixeiro viajante podem envolver a obtenção de soluções exatas usando, por exemplo, Programação em Lógica por Restrições (PESANT et al., 1998) ou soluções aproximadas usando, por exemplo, algoritmos genéticos (GREFENSTETTE et al., 1985) e (LARRAÑAGA et al., 1999). Os algoritmos de Otimização por Colônias de Formigas da Seção 2.4 são descritos em função do PCV.

2.2 Problema da Mochila

O Problema da Mochila (PM, *Knapsack Problem*, KP) é um problema clássico da otimização combinatória e da programação inteira que ocorre frequentemente na alocação de recursos com restrições. A formulação original do problema consiste em, dado um conjunto de itens com peso e valor determinados, selecionar um subconjunto de itens de modo a maximizar a soma dos valores mas mantendo a soma dos pesos inferior a um dado limite. O problema também é encontrado em uma série de variantes como o Problema da Mochila Real, Problema da Mochila Binária, Problema da Mochila Compartimentada e o Problema da Mochila Irrestrito (MARTELLO; TOTH, 1990).

O Problema da Mochila é um problema da classe de complexidade $\mathcal{NP}$-completo (PAPADIMITRIOU; STEIGLITZ, 1998), porém, há soluções eficientes para uma grande parte das instâncias práticas (PISINGER, 2005). Um estudo extensivo das variantes deste problema e de abordagens computacionais para seu tratamento é descrito em (MARTELLO; TOTH, 1990). As propriedades do PM também são usadas para descrever partes do Problema do Roteamento de Cabos em Painéis Elétricos nas Seções 3.1.1 e 3.5.1.

2.3 Problemas de Roteamento de Veículos

O Problema de Roteamento de Veículos (PRV ou VRP, *Vehicle Routing Problem*) é uma generalização do Problema do Caixeiro Viajante encontrada com frequência na área da pesquisa operacional. A definição original do problema, conforme (DANTZIG; RAMSER, 1959), descreve o processo de planejar a rota de veículo de carga com capacidade limitada que, partindo de uma dada origem, efetue entregas em um conjunto de locais de destino respeitando as restrições do problema (capacidade e carga específica para cada destino) a um custo mínimo (expresso em distância percorrida ou custo monetário do trajeto). Problemas de roteamento de veículos são estudados extensivamente há décadas e dezenas de variações são descritas na literatura.

A representação exata do espaço de busca de um PRV depende das características da variante desejada mas, para o caso geral, pode-se adotar um grafo valorado onde os nós representam os pontos a visitar e as arestas representam as ruas disponíveis, com seus pesos indicando custos (distâncias ou outros fatores adotados). Uma solução para um problema de roteamento de veículos é dada por um conjunto de caminhos representando as rotas traçadas pelo veículo. A Figura 2.1 exibe uma instância de um PRV com as possíveis soluções ligando o nó de origem (C0) aos nós de destino (C1 . . . C5).

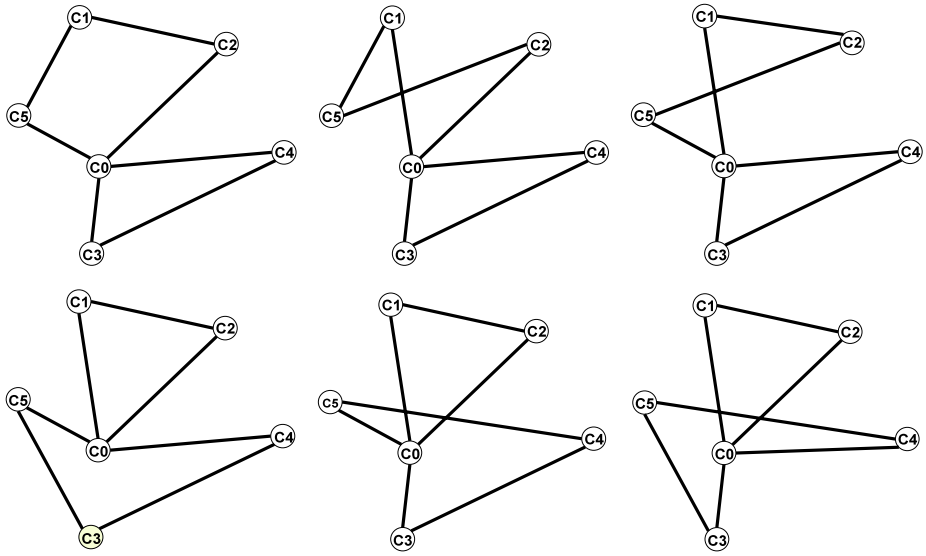


Figura 2.1: Exemplo de instância de um problema de roteamento de veículos. Reproduzido de (OLIVEIRA; VASCONCELOS, 2008).

As diferenças entre as muitas variações do PRV existentes na literatura podem ser ordenadas pelas seguintes propriedades:

- **Problema dinâmico ou estático.** Em um problema estático todos os dados relevantes são imutáveis e definidos previamente, enquanto em um problema dinâmico certas características estão sujeitas a mudança durante o atendimento como, por exemplo, a alteração no custo de determinada aresta do grafo de solução em função das condições de trânsito ou mudanças na lista de entregas após a partida dos veículos. Um exemplo de problema dinâmico é descrito em (BENT; HENTENRYCK, 2004);

- **Trajeto aberto ou fechado.** Em problemas do tipo *aberto*, não há necessidade de o veículo retornar ao ponto de origem, o que é exigido nos problemas do tipo *fechado*. Uma aplicação para problemas do tipo aberto é descrita em (PISINGER; RØPKE, 2007);

- **Número e tipo dos veículos.** Variações do problema podem exigir um ou mais veículos e, neste segundo caso, ainda é possível que os veículos sejam idênticos ou possuam características distintas que devem ser consideradas para a solução (adaptação a determinado tipo de carga, capacidade, tipo de serviço prestado, consumo de combustível, custo, etc.). Exemplos são descritos em (KRAJCAR et al., 1995) e (PESANT; GENDREAU; ROUSSEAU, 1997);

- **Existência de restrições temporais.** Em algumas versões do problema, o atendimento a determinado cliente só pode ocorrer durante períodos de tempo específicos. Estas janelas de tempo podem ser mandatórias, quando não é possível atender ao cliente fora do período especificado, ou desejáveis, quando o atendimento fora da janela temporal implica em mudança no custo da solução. Um exemplo de problema com janelas de tempo é descrito em (OLIVEIRA; VASCONCELOS, 2008);

- **Número de origens.** De acordo com a necessidade, variantes podem ter uma ou mais origens em diferentes posições do grafo de soluções e veículos e cargas podem estar posicionadas em origens distintas no início da resolução. Em (CORDEAU; GENDREAU; LAPORTE, 1997) descreve-se uma solução para uma variante com múltiplas origens;

- **Carga e limitação de capacidade de carga do veículo.** Determinadas variantes do problema podem desconsiderar limitações na quantidade de carga transportada pelo veículo ou não envolver o conceito de carga como, por exemplo, na prestação de serviços que não exigem insumos. Exemplos em (CORDEAU; LAPORTE, 2003) e (LETCHFORD; LYSGAARD; EGLESE, 2007);

- **Operações de coleta de carga.** Em algumas variantes o veículo deve coletar uma determinada carga em um cliente específico e, eventualmente, entregá-la para outro. É o caso do transporte de passageiros sob demanda, como descrito em (CORDEAU; LAPORTE, 2003).

Estas características são, normalmente, combinadas de acordo com a necessidade. Por exemplo, em (LETCHFORD; LYSGAARD; EGLESE, 2007), descreve-se um problema de roteamento de veículos capacitados em caminhos abertos. Diversas técnicas podem ser aplicadas na solução de Problemas de Roteamento de Veículos como, por exemplo, Programação Linear (DANTZIG; RAMSER, 1959), *branch-and-cut* (LETCHFORD; LYSGAARD; EGLESE, 2007), Programação por Restrições (PESANT; GENDREAU; ROUSSEAU, 1997) e (SHAW, 1998), Algoritmos

Genéticos (KRAJCAR et al., 1995), Busca Tabu (CORDEAU; GENDREAU; LAPORTE, 1997) e Otimização por Colônias de Formigas (RIZZOLI et al., 2007).

2.4 Otimização por Colônias de Formigas

A otimização por colônias de formigas (*ant colony optimization* – ACO) é uma meta-heurística de otimização inspirada no comportamento das formigas na natureza. Sua primeira aplicação na solução de problemas computacionais remonta aos trabalhos de (MANIEZZO et al., 1991; COLORNI; DORIGO; MANIEZZO, 1992) no início da década de 1990 e deriva do processo de busca do menor caminho entre um formigueiro e uma fonte de comida auxiliada pela deposição de feromônio no ambiente. Neste caso específico, é notável que um comportamento altamente estruturado emerge da interação entre muitos agentes cooperativos simples que executam ações especializadas, possuem conhecimento limitado do espaço de busca – as formigas estudadas por (GOSS et al., 1989) são praticamente cegas –, comunicam-se principalmente de forma indireta e carecem de uma coordenação ou organização centrais.

2.4.1 Colônias de Formigas Naturais

A literatura apresenta diversas análises do processo de busca de alimento por formigas e suas estratégias para a solução dos problemas inerentes à atividade (DENEUBOURG; PASTEELS; VERHAEGHE, 1983). Um exemplo particularmente interessante do comportamento das formigas na busca de caminhos mínimos é dado pelo experimento da ponte dupla (Figura 2.2) descrito em (GOSS et al., 1989). Este experimento é composto por duas plataformas, uma contendo o formigueiro e outra com uma fonte de comida, conectadas por uma ponte composta por dois segmentos com o formato representado na Figura 2.2a. Observou-se que, nos momentos inciais, a distribuição das formigas é essencialmente aleatória (Figura 2.2b) porém esta rapidamente converge para o caminho mais curto (Figura 2.2c).

Este comportamento é explicado pela comunicação indireta entre os insetos do experimento: enquanto caminham entre o ninho e a comida, as formigas depositam um marcador químico (feromônio) que evaporará gradualmente se não for reposto. Ao atingir uma das bifurcações, cada formiga segue uma direção escolhida probabilisticamente em função da quantidade de feromônio existente em cada braço do caminho. As formigas que seguem pelo caminho mais curto atingem seu objetivo antes das que seguem pelo caminho mais logo, começando sua viagem de retorno

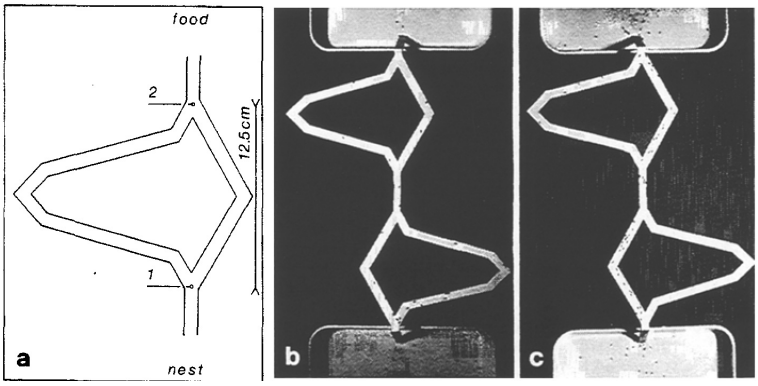


Figura 2.2: Experimento da ponte dupla, com um diagrama parcial das pontes (a) e distribuição das formigas após 4 (b) e 8 (c) minutos. Reproduzido de (GOSS et al., 1989).

mais cedo e repondo o feromônio evaporado. Consequentemente, com o passar do tempo, o caminho mais curto receberá uma quantidade maior de feromônio e atrairá mais formigas.

O formigueiro possui, portanto, um comportamento autocatalítico, representado pela realimentação positiva dos melhores caminhos, e um mecanismo de descarte ou "esquecimento" dos caminhos menos promissores, representado pela evaporação do feromônio.

A eficiência na adaptação a certas mudanças ambientais, como a obstrução de um caminho, é outra característica notável das colônias de formigas da Natureza. Este comportamento é exemplificado pelo experimento da Figura 2.3, descrito em (MANIEZZO et al., 1991), no qual uma trilha de formigas (A-E) é interrompida por um obstáculo, obrigando as formigas a escolherem um dos caminhos B-C-D ou B-H-D como alternativa ao caminho interrompido B-D. A probabilidade de um caminho específico ser escolhido é ponderada pela intensidade da trilha de feromônio existente, que é nula no momento da inserção do obstáculo, resultando em uma escolha aproximadamente aleatória. As formigas que seguem pelo caminho mais curto B-C-D atingem o ponto D antes das formigas que seguem pelo caminho mais longo B-H-D. Consequentemente, as formigas que retornam do ponto E encontram uma trilha de feromônio mais intensa no caminho mais curto, escolhem-no e depositam ainda mais feromônio, tornando-o ainda mais atrativo para as demais formigas. Como resultado, as formigas convergem e seguem pelo caminho curto.

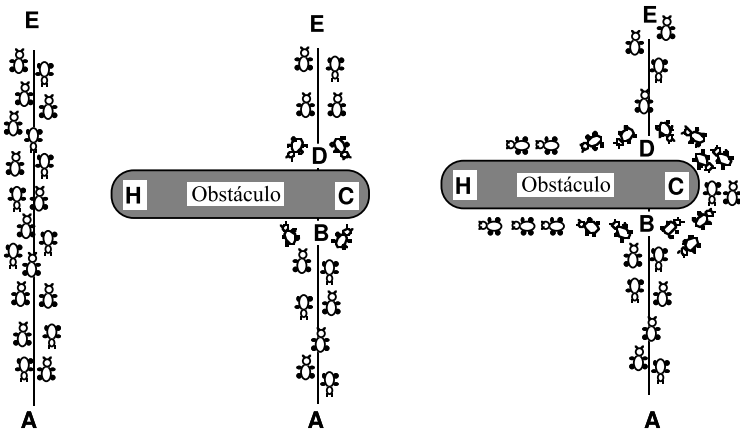


Figura 2.3: Comportamento das formigas ao desviar de um obstáculo. Adaptado de (DORIGO; MANIEZZO; COLORNI, 1996).

2.4.2 Colônias de Formigas Artificiais

Diversos modelos foram projetados para traduzir o comportamento das formigas reais para sistemas computacionais (ver Seções 2.4.3 e 2.4.4), mas todos compartilham algumas características comuns. Nestes modelos, as **formigas artificiais** são agentes computacionais simples que trabalham cooperativamente e comunicam-se depositando **trilhas de feromônio artificiais** em um ambiente artificial que representa o espaço de busca do problema. Conforme (DORIGO; CARO; GAMBARDELLA, 1999), algumas características que diferenciam as formigas artificiais das formigas naturais são:

- formigas artificiais existem em um mundo discreto e seus movimentos consistem de transações entre estados discretos;
- formigas artificiais possuem um estado interno com a memória das suas ações passadas;
- a quantidade de feromônio depositada por uma formiga artificial depende da qualidade da solução encontrada;
- o momento da deposição de feromônio varia de acordo com o problema. Algumas implementações só depositam feromônio quando uma solução for encontrada;
- o algoritmo pode ser complementado com buscas adicionais, inexistentes em formigas reais, como *look ahead*, otimização local e *backtracking*.

Os problemas tratáveis pela Otimização por Colônias de Formigas devem ser conversíveis para problemas de busca de caminhos mínimos restritos ou, como descreve-se em (CORDÓN; HERRERA; STÜTZLE, 2002), um problema de otimização por colônia de formigas possui um conjunto de componentes $N = \{n_1, n_2, \ldots, n_n\}$ mapeados para nós de um grafo de construção $G = (N, A)$, onde o conjunto de arestas $A = \{a_1, a_2, \ldots, a_n\}$ representa as possíveis transições entre os componentes. Um estado do problema é representado por um caminho δ sobre o grafo G e uma solução para o problema é um caminho que atenda todas as restrições impostas pelo problema. Soluções tem um custo que pode, eventualmente, ser associado a estados ou a transições. Componentes e transições podem ser associados a trilhas de feromônio τ representando a memória de longo prazo da solução e a valores heurísticos η representando alguma informação heurística disponível sobre o problema.

A operação genérica de um algoritmo de otimização por colônia de formigas é representada no Algoritmo 2.1. Variações deste algoritmo são possíveis, dependendo do modelo, problema e implementação, mas o processo geral segue as regras descritas a seguir.

Algoritmo 2.1 Otimização por colônias de formigas genérica

```
1 Preparar grafo de construção e formigas;
2 Enquanto os critérios de término não forem satisfeitos faça:
3     Movimentar formigas;
4     Atualizar feromônio;
5     Executar ações automáticas; /* opcional */
6 fim-enquanto
```

A preparação do algoritmo envolve a criação do grafo de construção, definição das quantidades iniciais de feromônio e preparação das formigas. Parâmetros como a quantidade inicial de feromônio, quantidade de formigas, etc. dependem do problema e da versão específica do algoritmo.

A etapa de movimentação das formigas consiste em deslocar cada formiga da população para um novo estado do problema em tempo discreto. A probabilidade de uma formiga mover-se para um estado específico é ponderada pela quantidade de feromônio τ do estado de destino em função e da atratividade η deste estado calculada em função da informação (possivelmente heurística) disponível. A influência exata de τ e η sobre a movimentação das formigas varia em função da versão específica do algoritmo adotado. Em algumas versões do ACO, as formigas também depositam feromônio nesta etapa.

A etapa de atualização de feromônio consiste em "evaporar" o feromônio das trilhas, ou seja, reduzir sua concentração em uma razão pré-estabelecida. Este processo é fundamental para evitar a convergência prematura do algoritmo para uma região sub-ótima do espaço de busca.

As ações automáticas (também conhecidas como *daemon actions* na literatura inglesa) são operações opcionais capazes de redirecionar o processo de busca através de informações adicionais não disponíveis para as formigas. Estas ações permitem, por exemplo, hibridizar o algoritmo através da deposição de feromônio adicional em função dos resultados obtidos com outro algoritmo de busca. A aplicação desta etapa é dependente do problema.

Estas etapas são repetidas enquanto uma ou mais condições de término não forem satisfeitas. Tais condições são dependentes do problema e, como exemplo, pode-se citar a obtenção de uma solução com um custo conhecido, a estagnação da solução por um certo número de iterações ou o alcance de um tempo limite de execução do programa.

Há diversos algoritmos de otimização por colônias de formigas que diferem pela implementação dos processos não definidos pela versão genérica e pela eficiência na solução de problemas específicos. Dois destes algoritmos, descritos a seguir, são particularmente interessantes para a implementação de soluções para o Problema do Roteamento de Cabos em Painéis Elétricos; Entre outros algoritmos, não descritos, também cita-se o *Elitist Ant System*, Ant-Q (GAMBARDELLA; DORIGO, 1995), *Ant Colony System* (DORIGO; GAMBARDELLA, 1997), *Rank-based Ant System* (BULLNHEIMER; HARTL; STRAUSS, 1997), ANTS (MANIEZZO, 1999) e *Best-worst Ant System* (CORDÓN; VIANA; HERRERA, 2002).

2.4.3 *Ant System* (AS)

O *Ant System* foi o primeiro sistema de otimização baseado em colônia de formigas divulgado na literatura, introduzido pela pesquisa pioneira de Marco Dorigo e outros (COLORNI; DORIGO; MANIEZZO, 1992), para a solução do problema do caixeiro viajante. Embora a descrição inicial do modelo foque especificamente neste problema, é possível adaptá-lo para outros problemas equivalentes.

As formigas definidas para o AS possuem uma memória capaz de armazenar a lista de cidades já visitadas e visão capaz de estimar a distância entre a cidade atual e a próxima cidade candidata. A probabilidade p_{ij} de escolha de uma determinada cidade de destino, entre as cidades

ainda não visitadas, é dada por

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{j=1}^{n} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}, \tag{2.1}$$

onde τ_{ij} é a intensidade da trilha de feromônio na aresta ij para um cidade e η_{ij} é a "visibilidade" da cidade, isto é, uma função heurística da viabilidade da cidade dada pelo inverso da distância ($\eta_{ij} = 1/d_{ij}$). α e β são dois parâmetros que permitem controlar a importância relativa da quantidade de feromônio e da visibilidade, respectivamente, na definição da probabilidade de visitação de determinada cidade.

O *Ant System* define três algoritmos distintos em função da estratégia de deposição de feromônio:

- *ant-density*, no qual a deposição é feita durante a movimentação das formigas e cada formiga deposita uma quantidade predeterminada de feromônio em cada aresta visitada;
- *ant-quantity*, no qual a deposição é feita durante a movimentação das formigas e cada formiga deposita uma quantidade Q/d_{ij} de feromônio, onde Q é um parâmetro do algoritmo e d_{ij} a distância para a próxima cidade (representada pelo comprimento da aresta); e
- *ant-cycle*, no qual a deposição é feita após a obtenção de uma solução pela formiga, que deposita uma quantidade Q/L_k de feromônio, onde Q é um parâmetro do algoritmo e L_k o comprimento total (custo) da solução obtida, ou seja, a soma dos comprimentos de todas as arestas usadas para compor a solução.

Nos três algoritmos propostos em (COLORNI; DORIGO; MANIEZZO, 1992), a evaporação do feromônio é feita pela multiplicação do valor da intensidade de trilha por um coeficiente de evaporação τ_{ev}, tal que $0 < \tau_{ev} < 1$. Esta etapa prevê o "esquecimento" gradual das soluções parciais pouco eficientes.

As avaliações de desempenho executadas com os três algoritmos apontam a superioridade do *ant-cycle*, de modo que este é frequentemente descrito na literatura como a versão "canônica" do *Ant System* (CORDÓN; HERRERA; STÜTZLE, 2002). Conforme (STÜTZLE; HOOS, 2000), os resultados obtidos pelo AS na solução do Problema do Caixeiro Viajante são encorajadores, mas seu desempenho ainda é inferior ao de outras abordagens contemporâneas para este mesmo problema, o que incentiva o desenvolvimento de variantes mais eficientes.

2.4.4 $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* ($\mathcal{MMAS}$)

O $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* é um algoritmo de otimização por colônia de formigas proposto em (STÜTZLE; HOOS, 1997) e derivado do *Ant System* para solução do problema do caixeiro viajante e problemas relacionados, cujas principais características são a limitação da intensidade das trilhas de feromônio a um intervalo predefinido e a deposição de feromônio feita apenas pela formiga com a melhor solução (da última iteração ou desde o início da busca). Tais alterações são introduzidas com o objetivo de melhorar o desempenho do AS combinando uma maior exploração das melhores soluções parciais encontradas com um mecanismo eficiente para evitar a convergência prematura da busca.

Conforme (STÜTZLE; HOOS, 2000), o uso de uma única solução para a atualização do feromônio é o mais importante meio de exploração do espaço de busca no $\mathcal{MMAS}$, pois componentes da solução que aparecem com maior frequência em boas soluções tornam-se mais atrativos para a construção de soluções futuras. O algoritmo permite que a deposição de feromônio ocorra em função da melhor solução encontrada desde o início da sua execução (*global best*) ou a melhor solução encontrada na iteração mais recente (*iteration best*) e a escolha cuidadosa de uma destas duas alternativas é necessária para o bom desempenho da otimização – a aplicação exclusiva da melhor solução global pode causar a convergência prematura para um ótimo local, um risco que é reduzido e substituído pela possibilidade de uma convergência mais lenta ao se aplicar apenas a melhor solução da última iteração. Estratégias mistas também são possíveis como, por exemplo, alternar a aplicação das duas estratégias ou controlar a aplicação em função de uma busca local.

A probabilidade de escolha de uma determinada cidade de destino segue as mesmas regras do AS (Equação 2.1) e a atualização da intensidade das trilhas de feromônio do instante t para o instante $t+1$, após a construção das soluções pelas formigas, é executada segundo a fórmula

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij} \tag{2.2}$$

onde ρ é a persistência da trilha de feromônio (tal que a evaporação τ_{ev} equivale a $1-\rho$) e $\Delta\tau_{ij}$ é a quantidade de feromônio depositada na aresta ij. Esta quantidade é dada por

$$\Delta\tau_{ij} = \begin{cases} Q/L_k & \text{Para a melhor solução} \\ 0 & \text{Para as demais soluções} \end{cases} \tag{2.3}$$

onde Q é uma constante definida para o problema e L_k é o comprimento total (custo) do caminho da melhor solução obtida. O novo valor $\tau_{ij}(t+1)$ da intensidade do feromônio é limitado entre

os valores mínimo e máximo representados, respectivamente, por dois parâmetros τ_{min} e τ_{max}.

A definição adequada dos limites de feromônio (τ_{min}, τ_{max}) é fundamental para o bom desempenho do $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*. Embora seja possível determinar estes parâmetros empiricamente, um método analítico descrito em (STÜTZLE; HOOS, 2000) fornece melhores resultados. A definição deste método introduz o conceito de *convergência* para o $\mathcal{MMAS}$: considera-se que uma instância convergiu quando, para cada nó do grafo de construção, há exatamente uma aresta com a quantidade τ_{max} de feromônio e todas as demais possuem a quantidade τ_{min}. Nesta condição, pode-se demonstrar[1] que τ_{max} terá o valor de

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{f\left(s_{opt}\right)} \tag{2.4}$$

onde ρ é a persistência da trilha de feromônio e $f\left(s_{opt}\right)$ é o custo (comprimento) de uma solução ótima. Como esta informação não é previamente conhecida em muitas aplicações práticas, pode-se substituí-la pela melhor solução global encontrada em uma execução inicial e atualizar os limites continuamente em função dos novos ótimos globais obtidos durante a execução. O cálculo do valor ótimo para τ_{min} exige algumas suposições a respeito da natureza do espaço de busca, sendo dado por

$$\tau_{min} = \frac{\tau_{max} \cdot \left(1 - \sqrt[n]{p_{best}}\right)}{(avg-1) \cdot \sqrt[n]{p_{best}}} \tag{2.5}$$

onde n é o número de cidades do problema, p_{best} é um fator representando a probabilidade média de uma formiga escolher a melhor aresta disponível (desconsideradas as informações heurísticas) e avg é o grau médio dos nós do grafo de construção durante a execução do algoritmo. A necessidade destas informações adicionais tende a dificultar a determinação analítica de τ_{min} em muitas aplicações práticas. Já em (CORDÓN; HERRERA; STÜTZLE, 2002) afirma-se especificamente que "para τ_{min} normalmente basta escolher algum fator constante inferior a τ_{max}."

O efeito dos demais parâmetros (taxa de evaporação da trilha de feromônio ρ e influências da intensidade da trilha de feromônio α e da função heurística β) sobre o desempenho do $\mathcal{MMAS}$ é análogo ao seu efeito sobre o *Ant System*. Uma análise detalhada sobre a influência destes parâmetros é descrita em (PELLEGRINI; FAVARETTO; MORETTI, 2006).

Na inicialização do algoritmo, a intensidade da trilha de feromônio nas arestas é definida para o valor máximo (τ_{max}). Este fato leva a um aumento da diversificação das soluções iniciais, quando as diferenças entre as trilhas de feromônio são pouco acentuadas, prevenindo a

[1] A demonstração desta convergência é descrita em (STÜTZLE; HOOS, 2000).

convergência prematura (CORDÓN; HERRERA; STÜTZLE, 2002).

Os testes comparativos entre o $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* e *Ant System* em instâncias padronizadas do Problema do Caixeiro Viajante extraídas da TSPLIB descritos em (STÜTZLE; HOOS, 2000) apontam para uma clara superioridade do $\mathcal{MMAS}$ perante os demais algoritmos, apontando-o como o algoritmo de otimização por colônia de formigas mais eficiente disponível para o PCV. Outra avaliação, no mesmo artigo, também aponta sua eficiência para o Problema da Atribuição Quadrática.

2.5 Algoritmos Genéticos

Algoritmos genéticos são algoritmos de busca e otimização probabilísticas inspirados na evolução das espécies biológicas, conforme os estudos realizados por Charles Darwin e Gregor Mendel no século XIX. Estudos sobre a aplicação da evolução das espécies na solução de problemas computacionais remontam aos anos 50, porém suas aplicações popularizaram-se apenas a partir de 1970. Algoritmos genéticos são uma especialização da área mais ampla da computação evolucionária (FOGEL, 2006).

Uma característica notável dos algoritmos genéticos é tratar o problema como uma *caixa preta* – o algoritmo modifica as variáveis de entrada e avalia a qualidade da solução expressa pela sua função de adaptabilidade (*fitness*), sem analisar explicitamente o relacionamento entre as variáveis. Esta característica permite, por exemplo, encontrar soluções de problemas para os quais não se conhece nenhuma solução analítica. É imprescindível, portanto, a existência de uma função de *fitness* que permita uma comparação qualitativa das soluções. Não é válida, por exemplo, uma função que retorne apenas os resultados "correto" ou "incorreto" para um problema de atribuição (JONG; SPEARS, 1989).

Em um algoritmo genético, uma população de soluções candidatas (conhecidas como indivíduos) evolui para gerar soluções melhores a cada iteração. O tamanho desta população depende das características do problema e deve ser suficiente para cobrir adequadamente o espaço de busca. Do mesmo modo que a evolução das espécies biológicas favorece os indivíduos melhor adaptados ao meio, o algoritmo genético favorece as soluções candidatas melhor adaptadas ao problema.

Os indivíduos de um algoritmo genético possuem, à semelhança de seus pares biológicos, fenótipo e genótipo. O fenótipo é composto pelas variáveis de problema e é codificado em um

genótipo passível de tratamento por operadores genéticos de seleção, mutação e *crossover*. O método de codificação adotado depende das características do problema: um exemplo típico é a chamada codificação binária, onde o genoma é um vetor de bits com trechos delimitados para cada variável. Esta codificação é adequada para problemas compostos por variáveis inteiras e, com algumas adaptações, pode ser aplicado para variáveis reais.

Entretanto, os algoritmos genéticos não estão limitados a um tipo de codificação e alguns problemas exigem codificações não-triviais. Este é o caso de problemas combinatórios, como o clássico Problema do Caixeiro Viajante. Uma análise dos codificações e representações para algoritmos genéticos aplicados ao PCV é apresentada em (LARRAÑAGA et al., 1999).

O Algoritmo 2.2 demonstra o formato básico de um algoritmo genético conforme (GOLDBERG, 1989). A execução inicia-se com a criação de uma população inicial de cromossomos aleatórios e prossegue com a avaliação dos indivíduos e a aplicação sucessiva das operações de seleção, *crossover* e mutação (descritas nas seções a seguir). A execução do algoritmo finaliza quando um dos critérios de término especificados for atingido. Critérios de término comumente adotados são a superação de um número limite de gerações, a estagnação da melhor solução por um número limite de gerações ou a obtenção de um indivíduo com *fitness* superior a um dado limite.

Algoritmo 2.2 Algoritmo genético canônico

```
1 Criar a população inicial com indivíduos definidos aleatoriamente;
2 Enquanto os critérios de término não forem atingidos faça:
3     Avaliar a função objetivo para os indivíduos;
4     Atribuir o fitness de cada indivíduo;
5     Selecionar os indivíduos para reprodução usando um operador de seleção;
6     Reproduzir os indivíduos usando os operadores de crossover e mutação;
7 fim-enquanto
8 Retornar o melhor indivíduo como solução;
```

A etapa de avaliação do indivíduo consiste em extrair os dados das variáveis da informação cromossômica (conversão genótipo-fenótipo) e aplicá-las à função objetivo. O resultado da função objetivo é normalizado (se necessário) e ponderado com eventuais penalidades por violações de restrições do problema para formar o valor da função de *fitness* do indivíduo.

O comportamento do algoritmo genético é regido pelos seguintes parâmetros:

- **Tamanho da população.** Representa o número de cromossomos na população avaliada. Este valor deve ser ponderado entre a exploração desejada do espaço de busca e a eficiência necessária. Um grande número de indivíduos aumentará a cobertura do espaço de busca, porém, trará um maior custo computacional para a solução;

- **Probabilidade de *crossover*.** Representa a probabilidade de um dado cromossomo ser selecionado para reprodução, caso atenda aos requisitos da operação de seleção. Como regra geral, usa-se valores elevados ($> 80\%$);

- **Probabilidade de mutação.** Representa a probabilidade de um cromossomo da população sofrer mutações pela aplicação de operador de mutação. Como regra geral, usa-se valores reduzidos ($< 5\%$).

Algoritmos genéticos tem sido adaptados e aplicados nas mais diversas áreas nas últimas décadas. Uma listagem não-exaustiva de aplicações na área de otimização de processos industriais inclui variações do problema do caixeiro viajante (GREFENSTETTE et al., 1985), problemas de roteamento de veículos (KRAJCAR et al., 1995), determinação de parâmetros de filtros (CASTILLO et al., 2001), diversas outras aplicações industriais (LOPES, 1999) e demais problemas da classe $\mathcal{NP}$-completo (JONG; SPEARS, 1989).

2.5.1 Operação de Seleção

A operação de seleção é utilizada para a escolha dos indivíduos mais aptos a reprodução em função do valor de *fitness*, de forma análoga ao processo de seleção natural nas espécies biológicas. Dentre os diversos operadores de seleção disponíveis, dois especialmente práticos são a Seleção por Roleta e a Seleção por Truncamento.

2.5.1.1 Seleção por Roleta

O operador de seleção por roleta segrega um indivíduo aleatório do conjunto probabilisticamente em função do seu *fitness*, de modo que indivíduos com *fitness* elevado tenham maior chance de seleção. Um exemplo de seleção por roleta é dado na Figura 2.4. A figura representa um conjunto de indivíduos $[a, b, c, d, e, f]$ e seus respectivos valores de *fitness* $[12, 5, 7, 3, 10, 5]$, cuja soma é 42. A cada execução do operador, gera-se um número aleatório $0 \leq r \leq 42$ que "aponta" para

o indivíduo escolhido. Caso mais de um indivíduo seja desejado, repete-se o processo com os indivíduos restantes.

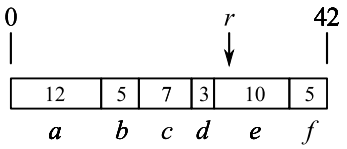


Figura 2.4: Operador de seleção por roleta

2.5.1.2 Seleção por Truncamento

O operador de seleção por truncamento (ou seleção elitista) segrega um subconjunto percentual dos melhores indivíduos do grupo em função direta do seu *fitness*. A cada execução do operador, ordena-se o conjunto de indivíduos pelo valor de *fitness* e seleciona-se os indivíduos superiores a um ponto de corte c. Um exemplo da sua aplicação é apresentado na Figura 2.5 para um conjunto de indivíduos $[a, b, c, d, e, f]$ cujos valores de *fitness* são, respectivamente, $[12, 5, 7, 3, 10, 5]$ (MÜHLENBEIN; SCHLIERKAMP-VOOSEN, 1993).

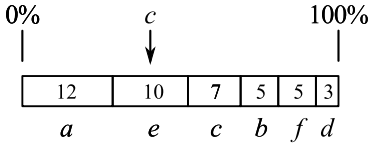


Figura 2.5: Operador de seleção por truncamento

2.5.2 Operação de *Crossover*

A operação de *crossover* (cruzamento ou recombinação) simula a reprodução sexuada de indivíduos na natureza, ou seja, a geração de novos indivíduos através da recombinação de material genético de dois indivíduos originais. Em um AG, este processo executa uma busca local. O processo exato de recombinação depende do tipo de codificação adotada para o algoritmo genético, mas o processo pode ser compreendido através dos operadores desenvolvidos para a codificação binária. Dois exemplos de operadores de *crossover* para esta codificação são exibidos na Figura 2.6, onde os cromossomos-pais A e B são seccionados em duas ou mais partes e recombinados, originando os cromossomos-filhos A' e B'.

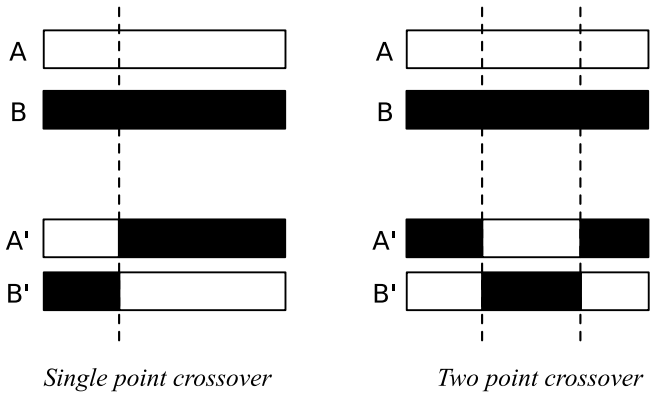


Figura 2.6: Operadores de *crossover* para codificação binária

As codificações adotadas para problemas combinatórios, a exemplo do PCV, exigem operadores de *crossover* específicos como *Cycle Crossover* (CX), *Order Crossover* (OX1), *Order Based Crossover* (OX2), *Position Based Crossover* (POS), *Heuristic Crossover*, *Genetic Edge Recombination Crossover* (ER), *Sorted Match Crossover*, *Maximal Preservative Crossover* (MPX), entre outros. Uma análise extensiva destes operadores está disponível em (LARRAÑAGA et al., 1999). O operador PMX é apresentado a seguir.

2.5.2.1 O Operador *Partially Mapped Crossover* (PMX)

O *Partially Mapped Crossover* (PMX) é um operador de recombinação específico para problemas combinatórios codificados na forma de listas de valores únicos. O Problema do Caixeiro Viajante é uma aplicação clássica deste tipo de codificação, onde cada gene do cromossomo representa uma cidade visitada.

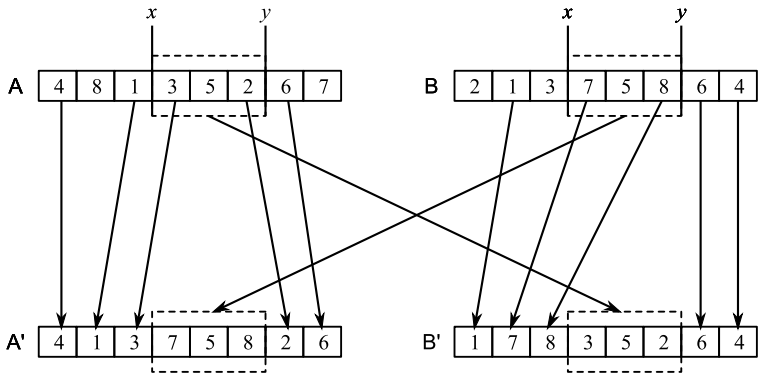


Figura 2.7: Operador PMX

A Figura 2.7 ilustra o processo de recombinação PMX. Inicialmente, escolhe-se dois valores

aleatórios x e y. Os cromossomos-pais A e B são seccionados neste intervalo e os genes intermediários são transferidos para os cromossomos-filhos A' e B'. Em seguida, os genes dos pais são copiados para os filhos, com exceção dos genes já presentes nos intervalos $[x - y]$. Como resultado, os cromossomos-filhos herdam partes da sequência dos pais, mas eventuais inconsistências são corrigidas.

2.5.3 Operação de Mutação

A operação de mutação simula falhas na cópia de genes entre cromossomos, um fato observado em organismos biológicos. Na Natureza, a mutação também é a principal fonte de diversidade genética em espécies de reprodução assexuada. Em um AG, este processo executa uma busca global. Outra função importante desta operação é combater a estagnação do algoritmo, restaurando sua diversidade genética em caso de convergência para um máximo local.

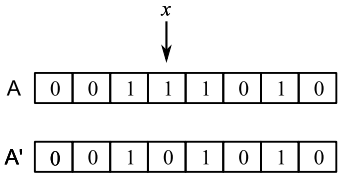


Figura 2.8: Operador de mutação para codificação binária

O funcionamento do operador de mutação para problemas com codificação binária é ilustrado na Figura 2.8. Dado um cromossomo A, seleciona-se um gene aleatório x e inverte-se o seu valor, gerando o cromossomo A' (LARRAÑAGA et al., 1999). Este processo pode ser executado antes ou depois da operação de *crossover*.

2.5.3.1 O Operador *Swap-Mutate* (SM)

O *Swap-Mutate* (SM) é um operador de mutação específico para problemas combinatórios codificados na forma de listas de valores únicos, como o PCV. O funcionamento do operador é ilustrado na Figura 2.9. Dado um cromossomo A, seleciona-se dois genes aleatórios x e y e troca-se os seus valores, resultando no cromossomo-filho A' (LARRAÑAGA et al., 1999).

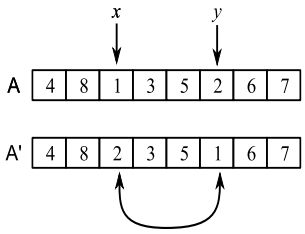


Figura 2.9: Operador *swap-mutate*

2.5.4 *Breeder Genetic Algorithm*

O *Breeder Genetic Algorithm* (BGA) é uma variante do algoritmo genético proposta em (MÜHLENBEIN; SCHLIERKAMP-VOOSEN, 1993) inspirada pela seleção *artificial* dos melhores indivíduos para reprodução, de forma análoga ao processo executado por criadores de animais de *pedigree*.

Há duas versões de BGA: o simples e o BGA distribuído (DBGA). Neste segundo caso, avalia-se um conjunto de instâncias do BGAs paralelamente com a troca eventual dos melhores indivíduos entre as instâncias, de modo similar ao compartilhamento dos melhores exemplares de uma determinada raça de animal entre criadores distintos.

As características notáveis do BGA são o uso da seleção por truncamento (Seção 2.5.1.2) e o controle efetivo sobre a qualidade da população através de regras de aceitação que permitem o descarte de cromossomos de baixa qualidade.

2.6 Resumo do Capítulo

A revisão da literatura existente indica que a Otimização por Colônias de Formigas, especialmente o $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*, fornece soluções eficientes para o Problema do Caixeiro Viajante e suas variantes. Estas técnicas também podem ser aplicadas para várias classes de problemas de roteamento de veículos e problemas correlatos. Com codificações e operadores específicos, também é possível usar Algoritmos Genéticos para a solução destes mesmos problemas.

3 Problema do Roteamento de Cabos em Painéis Elétricos

Um painel industrial é um equipamento usado para abrigar componentes de sistemas de automação, notavelmente componentes elétricos que precisam ser conectados por fios e cabos. Instalações industriais de médio e grande porte costumam ter várias dezenas ou centenas destes painéis, cada um abrigando dezenas de componentes elétricos interligados por centenas ou milhares de cabos. A Figura 3.1 apresenta um exemplo de painel elétrico. Os cabos usados para conexão destes equipamentos são abrigados em canaletas, eletrodutos e outros conduítes organizados na forma de um grafo. A disposição correta dos cabos nos conduítes afeta a estética, o custo e a organização das instalações industriais.

Nesse contexto, denomina-se Problema do Roteamento de Cabos em Painéis Elétricos (PRCPE) o problema de determinar um caminho para cada cabo do painel, conectando os componentes envolvidos ao menor custo possível e sem extrapolar o limite de espaço disponível nos conduítes.

Figura 3.1: Componentes internos de um painel. Fonte: WEG Automação S.A.

3.1 Definição Formal do Problema

Para fins de descrição, define-se um painel como composto por um número arbitrário de componentes (contatores, relês de sobrecarga, fusíveis, CLPs, etc.) contendo um número também arbitrário de terminais de conexão em uma dada posição física e um grupo de conduítes para a passagem dos cabos. Matematicamente, esse painel P_n, é representado pela tupla

$$P_n = (L_t, L_c, L_i) \tag{3.1}$$

onde L_t é a lista de terminais de ligação, L_c é a lista de conduítes e L_i é lista de interligações. O desenho esquemático de um painel simplificado pode ser visto na Figura 3.2, representando alguns componentes (KA1, KA2, KA3, Q1 e A10), canaletas e estrutura de suporte. A posição física dos componentes e terminais é definida pelo projetista segundo critérios de projeto e não é sujeita a mudanças pelo processo de roteamento. Instalações industriais de grande porte normalmente possuem vários painéis acoplados, que formam um conjunto com dezenas ou centenas de componentes interligados por milhares de cabos; o modelo apresentado considera apenas a topologia do painel e a organização dos conduítes, logo, continua válido para esta situação e é possível especificar vários painéis no mesmo modelo.

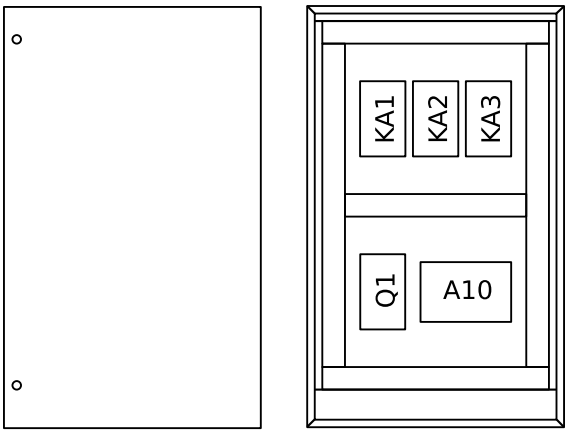


Figura 3.2: Um painel simplificado

Uma interligação I_n é um conjunto de terminais eletricamente equivalentes, sob o mesmo nível de potencial elétrico, e é definida por

$$I_n = (L_t, e_i, s_e, c) \tag{3.2}$$

onde L_t é a lista de terminais conectados, e_i é a especificação elétrica do cabo usado (bitola, cor, tensão de isolação, máxima temperatura de trabalho, classe de encordoamento, etc.), s_e é a área da seção transversal externa à isolação do cabo e c é o custo por unidade de comprimento do cabo. Os valores de e_i são importantes do ponto de vista elétrico, mas não são usados no processo de roteamento.

Cada interligação dá origem a um ou mais cabos, necessários para conectar eletricamente os terminais envolvidos. Um cabo w é dado por

$$w_n = (t_i, t_f, e_i, s_e, c) \tag{3.3}$$

onde t_i é o terminal inicial, t_f é o terminal final, e_i é a especificação do cabo, s_e é a área da seção transversal externa à isolação do cabo e c é o custo por unidade de comprimento do cabo.

Cada cabo conecta exatamente dois terminais e, por critérios de projeto, nenhum terminal pode ter mais que dois cabos conectados. Isso é válido para terminais de uma mesma lista de interligação, conectados na forma de uma *daisy-chain* ou sequência de *jumpers*, para os quais deseja-se que o algoritmo determine a melhor sequência de conexão, uma abordagem usual para circuitos de comando e cargas de baixa potência. Há exceções, como em barras de distribuição e em alguns circuitos de força, onde espera-se a presença de mais de dois cabos por terminal e, para permiti-las, restringe-se a limitação de dois cabos a terminais de uma mesma interligação, de modo que o projetista possa especificar estas exceções usando interligações distintas com dois terminais cada. Dados estes critérios, o número q_{cabos} de cabos necessário para executar uma ligação envolvendo q_{term} terminais é dado por

$$q_{cabos} = q_{term} - 1 \tag{3.4}$$

portanto, a interligação N conectando três terminais de componentes da Figura 3.3 pode ser executada com dois cabos.

A ordem sob a qual os terminais de uma interligação são conectados é particularmente importante, pois permite reduzir as distâncias percorridas pelos cabos. Por análise combinatória, sabe-se que há $q_{term}!$ permutações para a lista de terminais e que metade dessas permutações são inversões de outras já enumeradas. Quando essas permutações representam a ordem de conexão dos terminais, não há qualquer distinção do ponto de vista elétrico ou mecânico entre uma ligação e sua duplicata invertida. Deduz-se, portanto, que há $\frac{q_{term}!}{2}$ maneiras distintas de ordenar os terminais de cada interligação. Isto também significa que o número de possíveis sequências de conexão cresce exponencialmente em função do número de terminais interligados.

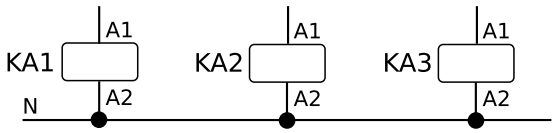


Figura 3.3: Interligação entre três terminais de três componentes distintos

Um terminal é definido por

$$T_n = (n_c, n_t, P_{xyz}) \tag{3.5}$$

onde n_c é a identificação do componente, n_t é a identificação do terminal conectado e P_{xyz} são as coordenadas do terminal no espaço tridimensional do painel.

Os conduítes são segmentos retilíneos de materiais usados para alojar cabos no interior dos painéis, como eletrodutos, canaletas, eletrocalhas e outros. Neste trabalho, o termo "conduíte" é usado como uma generalização destes materiais. Um conduíte é definido por

$$C_n = (s_c, A_{xyz}, B_{xyz}, t_c) \tag{3.6}$$

onde s_c é a área da seção transversal efetivamente disponível para passagem dos cabos, considerando eventuais margens de segurança, A_{xyz} e B_{xyz} são, respectivamente, os pontos de início e fim do conduíte no espaço do painel e t_c indica o tipo do conduíte. Para fins de modelagem do problema, define-se dois tipos de conduítes:

Conduítes abertos. São aqueles que permitem a passagem de cabos através das suas laterais (canaletas, por exemplo), quando esse comportamento é aceitável no projeto. Na Figura 3.4, as canaletas horizontais, representadas em linhas tracejadas, são um exemplo desse tipo de conduíte.

Conduítes fechados. São aqueles que só permitem a passagem dos cabos pelas suas extremidades (eletrodutos, por exemplo) ou quando a passagem de cabos pelas laterais não é desejável segundo os critérios do projeto. As canaletas verticais da Figura 3.4 permitiriam a passagem de cabos pelas laterais, porém, como esse comportamento não é desejável, elas foram modeladas como conduítes fechados.

Conduítes curvos podem ser modelados através de uma sequência de conduítes retilíneos. Esta estratégia permite definir o comprimento do conduíte para a distância euclidiana tridimensional entre os pontos A_{xyz} e B_{xyz}.

O primeiro e o último lance de cada cabo podem passar pelas laterais de um conduíte aberto, se não houver alguma extremidade de conduíte mais próxima, dando origem a dois pontos denominados *pontos de entrada* e representados na Figura 3.5 por $P1$ e $P2$, que devem ser considerados no cálculo do comprimento total do cabo. O problema da determinação desse ponto pode ser especificado geometricamente por: "Dado o segmento de reta $\overline{AB}$ representando um conduíte entre os pontos A e B e um ponto C representando o terminal de ligação, encontrar o ponto de entrada D sobre $\overline{AB}$ tal que o comprimento do segmento $\overline{CD}$ seja mínimo." Esse processo é repetido para todos os conduítes, localizando-se o ponto de entrada mais próximo do terminal em questão.

Pode-se representar os conduítes de um painel como arestas de um grafo valorado interligando nós atribuídos arbitrariamente para manter a mesma topologia existente no painel, como visto na Figura 3.4. Os conduítes possuem um espaço interno limitado, portanto, a quantidade de cabos que pode transitar por uma aresta do grafo é limitada em função da soma das áreas das seções transversais do cabos envolvidos. Essa limitação é representada por um valor de fluxo máximo associado a cada aresta do grafo. Considera-se como saturado o conduíte que, dada esta limitação, não pode receber novos cabos.

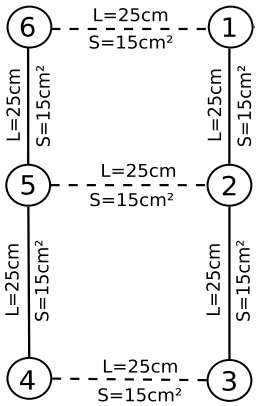


Figura 3.4: Grafo dos conduítes do painel da Figura 3.2

Uma rota é, para fins de definição, uma sequência de nós e terminais que indica o caminho percorrido por um cabo entre o terminal de origem t_o e o terminal de destino t_d, ou seja

$$r_n = [t_o, n_1, n_2, \ldots, n_n, t_d] \tag{3.7}$$

onde n são os nós percorridos pelo cabo. Caso o cabo passe por conduítes do tipo *aberto*, os respectivos pontos de entrada também são considerados. Por exemplo, a rota percorrida pelo

cabo representado na Figura 3.5, considerando a numeração dos nós definida na Figura 3.4, é:

$$r_1 = [\text{A10:N}, \text{P2}, 3, 2, \text{P1}, \text{KA1:A2}]$$

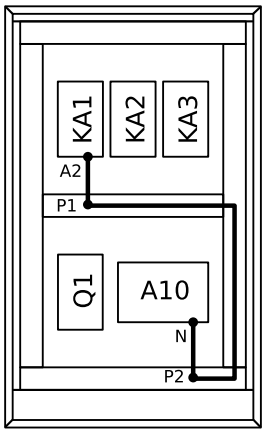


Figura 3.5: Rota de um cabo no painel

O comprimento de uma rota é dado pela soma do comprimento dos conduítes percorridos pelo cabo, considerando eventuais pontos de entrada. Ou seja, o comprimento da rota r_n, definida acima, é dado por

$$compr(r_m) = \sum_{i=1}^{m-1} dist(n_i, n_{i+1}) \tag{3.8}$$

Portanto, formalmente, o problema do roteamento de cabos consiste em determinar simultaneamente um conjunto de cabos $L_w = \{w_1, \ldots, w_m\}$ e um conjunto de rotas L_r, para cada interligação i_n pertencente a lista de interligações do painel, de forma a minimizar a função de custo

$$c_{total} = \sum_{j=1}^{n} c_{unit}(i_j) \times \sum_{k=1}^{m} compr(R(w_k)) \tag{3.9}$$

onde $c_{unit}(i_j)$ é o custo por unidade de comprimento do cabo usado para a ligação i_j e $R(w)$ é a função que associa um cabo w à uma rota entre todas as rotas possíveis para esse cabo.

Dado o conjunto de conduítes L_c do painel e um conjunto de cabos L_w passando por um conduíte $c \in L_c$, a função $R(w)$ deve respeitar a restrição

$$\sum sect(w) \leq s_c \forall w \in L_w, c \in L_c \tag{3.10}$$

onde $sect(w)$ é a área da seção transversal externa do cabo w e s_c é a área da seção transversal disponível no conduíte c.

3.1.1 Análise da Complexidade do Problema

Pode-se decompor o problema do roteamento de cabos em painéis em uma série de sub-problemas com diferentes graus de complexidade. Dois desses sub-problemas são identificáveis como fontes de complexidade:

- A definição da ordem de conexão dos terminais de uma interligação constitui uma variante do Problema do Caixeiro Viajante (ver Seção 2.1). O processo mais amplo de elaborar a lista de cabos e roteá-la no grafo de conduítes pode ser interpretado como um caso específico de Problema de Roteamento de Veículos (ver Seção 2.3). Há uma instância desse problema para cada interligação do painel.
- O espaço disponível nos conduítes reduz à medida que cabos são inseridos, o que levará à saturação de segmentos e alteração nas rotas, influindo no custo dos cabos. Pode-se entender essa propriedade como uma variante do problema da mochila (ver Seção 2.2) onde o custo da colocação de um item (representado pelo cabo) na mochila (representado pelo grafo de conduítes) varia à medida que o problema evolui.

Disto, tem-se que o problema descrito apresenta uma complexidade computacional equivalente ao Problema do Caixeiro Viajante, logo, pertence à classe dos problemas $\mathcal{NP}$-completos, e não possui solução trivial em termos de tempo de processamento e espaço em memória. A solução deste problema em tempos aceitáveis exige técnicas apuradas de otimização.

É válido observar que a remoção do limite de dois cabos por terminal de uma interligação transforma o Problema do Caixeiro Viajante no problema da geração de uma árvore geradora de peso mínimo, que pode ser tratado eficientemente em tempo linear (CORMEN et al., 2001). A solução gerada é, entretanto, imprópria para aplicação na fiação de um painel.

3.2 Tratamento de Conduítes Abertos

A existência de conduítes abertos em painéis origina considerações a respeito da *saturação parcial* de um conduíte, que ocorre quando alguns segmentos deste conduíte tornam-se saturados, mas outros ainda possuem espaço suficiente para mais cabos. A Figura 3.6 demonstra um exemplo de um conduíte parcialmente saturado.

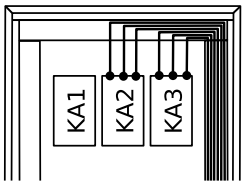


Figura 3.6: Saturação parcial de um conduíte.

Para tratar este problema, usa-se um processo adicional chamado *divisão de conduítes* que transforma todos os conduítes abertos em um conjunto de conduítes fechados delimitados pelos pontos de entrada dos cabos que os cruzam.

a. Ponto de entrada mais próximo Divide-se os conduítes exatamente sobre o ponto de entrada, como visto na Figura 3.7a. Esta estratégia é simples e dá resultados precisos quando ao comprimento dos cabos, mas pode gerar arestas excessivamente curtas, aumentando a complexidade do grafo e o tempo de resolução, sem grandes melhorias no resultado.

b. Ponto mais próximo com arredondamento Esta estratégia evita a criação de arestas curtas procurando por extremidades de conduítes a uma distância inferior a uma distância limite indicada pelo usuário. Se houver tal conduíte, usa-se seu ponto de entrada sem a execução de um novo seccionamento, como visto na Figura 3.7b. Esta estratégia fornece uma boa precisão, mas é mais lenta que a anterior, pois também exige a busca por extremidades de conduítes.

c. Seccionamento em distâncias constantes Esta estratégia secciona os conduítes em intervalos regulares com uma distância especificada pelo usuário, como visto na Figura 3.7c. esta é uma alternativa rápida, pois não executa buscas por arestas existentes ou cálculos de pontos de entrada, mas pode criar conduítes desnecessários, isto é, segmentos de conduítes de cujas extremidades não parte nenhum cabo (ver a última linha de conduítes da Figura 3.7c).

3.3 Abordagem Computacional

A definição apresentada na Seção 3.1 para o problema do roteamento de cabos em painéis dificulta a representação computacional do problema, dada a interdependência das operações de definição de rotas e seleção dos cabos para cada interligação.

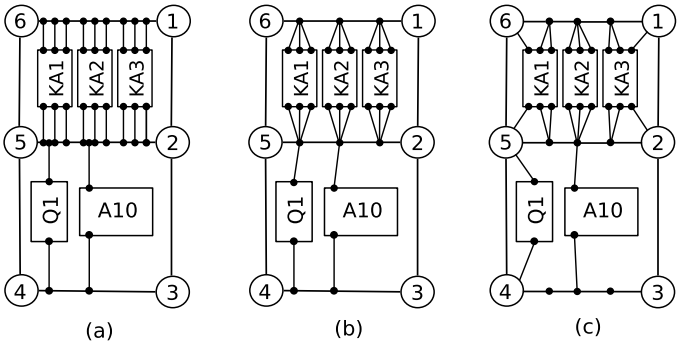


Figura 3.7: Estratégias para seccionamento de conduítes

Uma descrição alternativa para o problema, mais natural em termos computacionais, é: "dado um painel P com uma lista de interligações L_i e uma lista de conduítes L_c, inserir cada interligação no painel, gerando os cabos necessários, através da rota mais curta disponível em uma ordem tal que o custo final da solução seja mínimo".

Esta nova abordagem divide o problema em um problema de sequenciamento (determinação da sequência ideal de inserção de interligações no painel) e outro problema de roteamento (definição dos cabos e das rotas ótimas para cada interligação). A cada novo cabo inserido, atualiza-se o modelo do painel reduzindo o espaço ocupado pelo cabo do espaço disponível nos conduítes percorridos.

Esta abordagem difere sutilmente da abordagem descrita na Seção 3.1 quanto à determinação da otimalidade de uma solução candidata apresentada. Pode-se afirmar que a solução é ótima, independentemente da ordem de inserção, se nenhum cabo foi desviado da rota mais curta em função da saturação de conduítes. A afirmação contrária, porém, não é necessariamente verdadeira.

3.4 Trabalhos Anteriores e Problemas Relacionados

O Problema do Roteamento de Cabos em Painéis Elétricos, conforme descrito na Seção 3.1, não possui um paralelo na literatura. Há, porém, diversos problemas assemelhados que são descritos a seguir.

Um problema de roteamento de cabos em instalações prediais e sua respectiva solução através de algoritmos genéticos são descritos em (KLOSKE; SMITH, 1994). A configuração adotada para este problema é semelhante ao modelo simplificado descrito na Seção 3.5.1. Em (MA

et al., 2006) descreve-se um problema assemelhado, também resolvido através de Algoritmos Genéticos.

Há algumas semelhanças entre o roteamento de uma interligação no Problema do Roteamento de Cabos em Painéis Elétricos e a busca de caminhos em redes metabólicas descrita em (DOOMS; DEVILLE; DUPONT, 2005) e (PALACIOS; GEFFNER, 2002), que consiste em determinar um caminho mínimo restrito representado por uma sequência de reações químicas envolvendo, obrigatoriamente, um dado conjunto de moléculas. A rede metabólica é representada na forma de um grafo (Figura 3.8) e pode-se criar um paralelo entre este problema e a busca de um caminho mínimo para uma interligação elétrica passando, obrigatoriamente, por um dado conjunto de terminais. Uma abordagem para este problema usando Programação em Lógica por Restrições no domínio dos grafos é descrita em (VIEGAS; AZEVEDO, 2007). A aplicação desta técnica no Roteamento de Cabos em Painéis Elétricos foi estudada, porém, a inexistência de restrições propagáveis no preenchimento dos conduítes inviabiliza soluções eficientes. A inexistência destas restrições também inviabiliza a aplicação desta técnica em instâncias maiores do Problema do Caixeiro Viajante, como descrito em (CASEAU; LABURTHE, 1997).

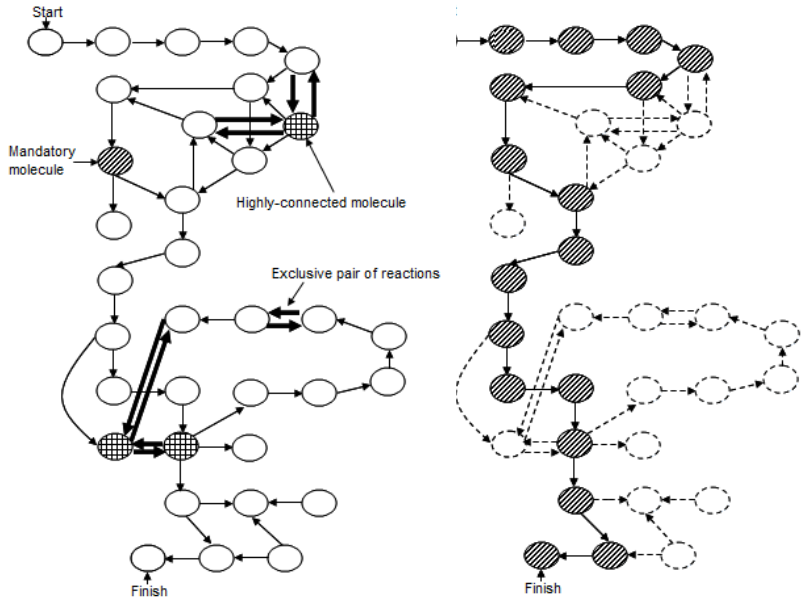


Figura 3.8: Problema das redes metabólicas. Fonte: (VIEGAS; AZEVEDO, 2007)

Os aplicativos CAE para a área elétrica EPLAN Cabinet (EPLAN, 2009) e E^3.series (ZUKEN, 2009) possuem rotinas para roteamento de cabos parcialmente semelhantes ao problema definido, porém, as técnicas empregadas não são descritas na literatura disponível publicamente.

A literatura também apresenta algumas classes de problemas de roteamento de cabos que podem ser confundidos com o Problema do Roteamento de Cabos em Painéis Elétricos, mas são fundamentalmente distintos. São exemplos: (1) o problema da construção de dutos de cabos descrito em (VASKO et al., 2002), cujo objetivo é a determinação de uma árvore geradora mínima restrita; (2) o problema do roteamento de cabos sobre estruturas descrito em (KABUL; GAYLE; LIN, 2007), que implica na simulação das propriedades mecânicas dos cabos e da sua interação com estruturas de suporte; e (3) o problema da geração de árvores de conduítes, como descrito em (CONRU, 1994).

3.5 Roteamento de Cabos Simplificado

O modelo descrito na Seção 3.1 permite soluções de alta qualidade, porém, sua elevada complexidade torna necessário o estudo de modelos simplificados que permitam soluções computacionais próximas à ótima em tempo reduzido. Dois destes modelos foram estudados.

3.5.1 Modelo Simplificado I

Um primeiro modelo simplificado do Problema do Roteamento de Cabos em Painéis Elétricos foi desenvolvido e descrito em (ITTNER; SÁ; SASSE, 2007) e (ITTNER; SÁ; SASSE, 2009). As principais mudanças em relação ao modelo completo, descrito na Seção 3.1, são:

- Efetuar o roteamento usando uma lista de cabos previamente definida, ao invés de usar a lista de interligações especificada na definição do problema. Esta simplificação dispensa o algoritmo de determinar as sequências de interligação e reduz enormemente a complexidade computacional do problema, porém exige a especificação dos cabos pelo usuário durante a elaboração do modelo para cada instância do problema, que é uma atividade onerosa e sujeita a erros, e impede o algoritmo de obter rotas mais curtas através de mudanças na ordem de conexão dos terminais de uma determinada interligação.
- Usar apenas conduítes do tipo *fechado*, com todos os cabos acessando-os pelas extremidades. Essa simplificação resume a localização dos pontos de entrada a uma busca pela extremidade de conduíte mais próxima ao terminal, feita em tempo linear[1], ao custo de distorções na solução final, exemplificadas na Figura 3.9.

[1]Essa busca linear possui complexidade $O(2n)$ para n conduítes. Uma otimização adicional, possível quando há conjuntos com vários painéis acoplados, mas não implementada na solução proposta, consiste em agrupar os

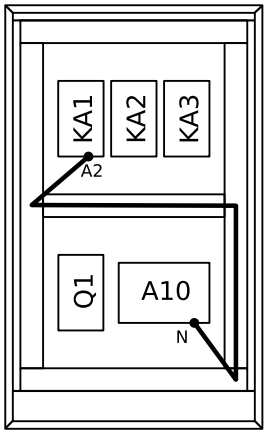


Figura 3.9: Rota da Figura 3.5 distorcida pela ausência dos pontos de entrada

Sem a complexidade causada pela geração da lista de cabos, uma solução para o modelo simplificado pode trabalhar unicamente com a atribuição de rotas para cada cabo. Um cabo percorrerá a rota mais curta entre os terminais de origem e destino, exceto se essa rota inviabilizar a existência de rotas mais econômicas para outros cabos por saturação dos conduítes.

Descrito desse modo, o problema possui algumas semelhanças com o clássico problema da mochila: há um recipiente de capacidade limitada (o grafo das canaletas) e diferentes objetos (cabos) com custos distintos que precisam ser inseridos com aproveitamento ótimo do espaço disponível. Há, porém, uma dificuldade adicional, inexistente no problema da mochila: o custo de um cabo varia a medida que o problema evolui, ou seja, a medida que novos objetos são inseridos, devido a saturação das canaletas.

Uma abordagem inicial para este problema usando um *backtrack* cronológico e uma técnica de *first-fail* (ver Algoritmo 3.1) produziu bons resultados quando aplicada a uma instância real do problema (ITTNER; SÁ; SASSE, 2007). Pelo algoritmo proposto, cada cabo é roteado pelo caminho mais curto disponível no grafo dos conduítes que, em seguida, tem seu espaço disponível reduzido em função da área da seção transversal externa do cabo; esse processo é chamado de "encolhimento do grafo".

conduítes e terminais por painel e restringir a busca aos conduítes localizados no mesmo painel do terminal.

Algoritmo 3.1 Roteamento Simplificado I

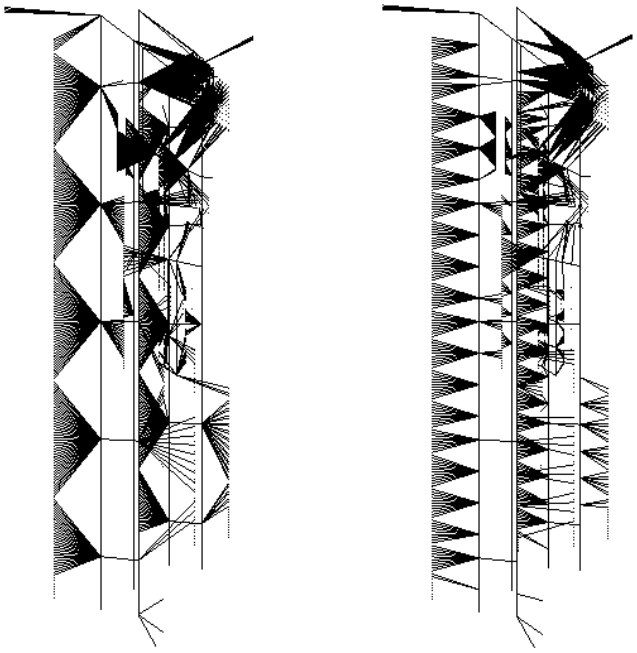
1 Dados uma lista de cabos L_w, uma lista de terminais L_t, uma lista de conduítes L_c e uma lista de nós L_n;
2 Classificar L_w em ordem decrescente em função do custo do cabo por unidade de comprimento;
3 **Enquanto** nenhuma solução válida for encontrada **faça**:
4 **Para** cada cabo $w \in L_w$ **faça**:
5 Determinar os nós iniciais e finais no grafo de conduítes, tal que as distâncias $dist(w_{inicial}, n_{inicial})$ e $dist(w_{final}, n_{final})$ sejam mínimas;
6 Determinar o conjunto L'_c com os conduítes de L_c com espaço interno livre suficiente para a passagem do cabo w;
7 Determinar o caminho mínimo r_w entre os nós $n_{inicial}$ e n_{final} de L'_w;
8 Atualizar o espaço disponível nos conduítes de L_w de acordo com r_w;
9 Determinar o custo da ligação, tal que:
 $c_{lig} = c_{unit} \times (dist(w_{inicial}, n_{inicial}) + compr(r_w) + dist(w_{final}, n_{final}))$
10 **fim-para**
11 Calcular o custo total da solução obtida;
12 Finalizar se a solução for válida; Caso contrário, permutar L_w;
13 **fim-enquanto**

3.5.2 Modelo Simplificado II

O segundo modelo simplificado para o Problema do Roteamento de Cabos em Painéis Elétricos remove a restrição à inexistência de conduítes do tipo aberto, mas mantém a limitação do roteamento a cabos previamente definidos. Um estudo detalhado deste modelo é descrito em (ITTNER; SÁ; SASSE, 2008).

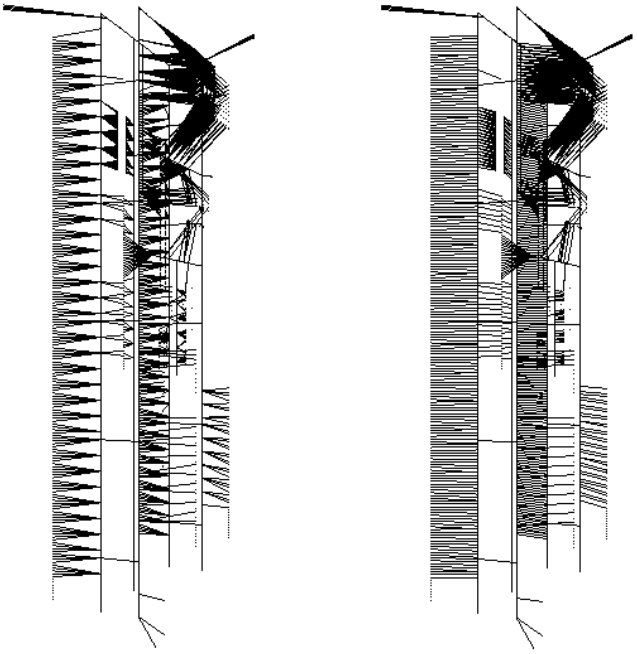
Para a solução deste problema, adiciona-se uma etapa de pré-processamento responsável por tratar os conduítes abertos, aplicando o método de seccionamento em distâncias constantes descrito na Seção 3.2, uma mudança refletida no Algoritmo 3.2.

As soluções geradas por este modelo são superiores em qualidade às geradas pelo Modelo Simplificado I, dada a menor distorção nas rotas dos cabos. Um exemplo comparativo desta distorção, para diversos valores de seccionamento, é exibido na Figura 3.10. Deve-se notar, entretanto, que o número mais elevado de nós e arestas nos grafos gerados pela etapa de seccionamento de conduítes aumenta o tempo de processamento da solução.



(a) Modelo I

(b) Modelo II, 100 mm



(c) Modelo II, 50 mm

(d) Modelo II, 10 mm

Figura 3.10: Comparação entre a distorção das rotas provocada pelos Modelos Simplificados I e II, com diversos valores de seccionamento, para a mesma instância do PRCPE

Algoritmo 3.2 Roteamento Simplificado II

1 Dados a lista de cabos L_w, a lista de terminais L_t, a lista de conduítes L_c, a lista de nós L_n e um comprimento mínimo de um conduíte l_{min};

2 **Para** cada conduíte aberto $c \in L_c$ com $comprimento(c) > l_{min}$ **faça**:

3 Criar um conjunto de conduítes fechados L'_c tal que $comprimento(c') \leq l_{min} \forall c' \in L'_c$ e a concatenação de $[L'_{c_1} \ldots L'_{c_n}] = c$ e inseri-lo em L_c;

4 Remover c de L_c;

5 **fim-para**

6 Classificar L_w em ordem decrescente de acordo com o custo unitário por unidade de comprimento do cabo;

7 **Enquanto** nenhuma solução válida for encontrada **faça**:

8 **Para** cada cabo $w \in L_w$ **faça**:

9 Encontrar, no grafo, os nós de início e fim para os quais as distâncias $dist(w_{inicial}, n_{inicial})$ e $dist(w_{final}, n_{final})$ sejam mínimas;

10 Determinar o conjunto L'_c contendo os conduítes de L_c com espaço interno suficiente para o cabo w;

11 Encontrar o caminho mínimo r_w entre os nós $n_{inicial}$ e n_{final} de L'_c;

12 Atualizar o espaço disponível no conduíte L_w conforme r_w;

13 Determinar o custo da rota $c_{rota} = c_{unitário} \times (dist(w_{inicial}, n_{fim}) + comprimento(r_w) + dist(w_{inicial}, n_{final}))$

14 **fim-para**

15 Determinar o custo total da solução atual;

16 Finalizar o programa se uma solução for encontrada; Caso contrário, permutar L_w;

17 **fim-enquanto**

3.6 Resumo do Capítulo

O Problema do Roteamento de Cabos em Painéis Elétricos consiste em determinar o conjunto de cabos e rotas de cabos para a execução das ligações elétricas entre os componentes de um painel a um custo mínimo, respeitando as limitações e exigências inerentes ao processo. Os dados de entrada do problema são: os componentes e terminais conectados, os caminhos disponíveis para os cabos (usando uma definição generalizada de conduíte), a lista de interligações entre os terminais e os dados dos condutores usados.

A complexidade do processo deriva principalmente da necessidade de determinar a ordem ótima de conexão dos terminais (um tipo específico de Problema de Roteamento de Veículos) e da saturação dos conduítes (uma variação do Problema da Mochila). Conduítes do tipo aberto adicionam o problema da saturação parcial, que deve ser tratada por uma etapa de pré-processamento.

Uma formulação alternativa do problema, que permite dividi-lo em etapas de sequenciamento e roteamento, é usada como base para a implementação computacional descrita no Capítulo 4.

4 Algoritmos e Implementação

Este capítulo apresenta diversos algoritmos desenvolvidos para o tratamento computacional do Problema do Roteamento de Cabos em Painéis Elétricos, conforme definido no capítulo anterior. A Seção 4.1 descreve três algoritmos propostos para a etapa de inserção de interligações e a Seção 4.2 descreve três outros algoritmos desenvolvidos para a etapa de roteamento. A implementação destes algoritmos é descrita na Seção 4.3.

4.1 Algoritmos para a Etapa de Inserção de Interligações

A etapa de inserção de interligações consiste em determinar a sequência ótima de inserção de interligações no grafo de painel conforme a definição da Seção 3.3, chamar um dos algoritmos para solução da etapa de roteamento (Seção 4.2) para gerar a lista de cabos e rotas para cada interligação e, por fim, inserir os cabos gerados no grafo do painel, atualizando seus atributos. As três abordagens desenvolvidas para esta etapa são descritas nas Seções 4.1.1, 4.1.2 e 4.1.3.

4.1.1 Inserção Heurística Simples

O algoritmo de Inserção Heurística Simples (IHS) foi desenvolvido para tratar as configurações mais comuns encontradas em painéis elétricos industriais, ponderando a possibilidade de obtenção rápida de *boas* soluções nas instâncias mais comuns com a possibilidade de execução em tempo exponencial no pior caso. Este algoritmo pode ser classificado como uma aplicação de um *backtrack* cronológico. A primeira versão deste algoritmo foi desenvolvida para tratar as versões simplificadas do PRCPE descritas nas Seções 3.5.1 e 3.5.2.

O funcionamento do algoritmo pode ser resumido em: (1) classificar a lista de interligações em ordem decrescente em função do custo do cabo designado para cada interligação, (2) chamar um algoritmo da etapa de roteamento para determinar o caminho mínimo para este cabo conforme o estado atual do grafo do painel e (3) inserir os cabos gerados no grafo, reduzindo o espaço disponível nos conduítes conforme a área da seção transversal externa do cabo recém-inserido. A inserção das interligações subsequentes na fila de inserção é feita conforme este novo estado do grafo. Caso, em qualquer momento, não seja possível inserir algum cabo devido a falta de

espaço nos conduítes (saturação), o algoritmo descarta a solução atual, permuta a lista de cabos e reinicia o processo.

Algoritmo 4.1 Funções de permutação para inserção heurística simples

```
local function permute(list, i)
    if #list == i then
        -- Memoriza o estado da co-rotina e devolve o resultado
        coroutine.yield(list)
    else
        -- Permuta os elementos da lista recursivamente
        for j = i, #list do
            list[i], list[j] = list[j], list[i]
            permute(list, i+1)
            list[i], list[j] = list[j], list[i]
        end
    end
end

-- Cria um iterador para as permutações da lista encapsulando
-- chamadas à função 'permute' em uma co-rotina.
local function permutations(list)
    return coroutine.wrap(function()
        permute(list, 1)
    end)
end
```

A consequência mais notável da aplicação deste algoritmo é a atribuição das rotas mais curtas disponíveis às interligações que exigem os cabos mais caros, minimizando o custo da fiação do painel. O algoritmo, porém, não garante a obtenção da configuração ótima para o painel na presença de conduítes saturados. A permutação da lista de interligações só é feita caso a saturação de um ou mais conduítes impeça a obtenção de uma rota, ainda que não-ótima, entre dois terminais. Logo, a aplicação deste algoritmo é justificada pela baixa probabilidade de saturação dos conduítes nas instâncias mais comuns do Problema do Roteamento de Cabos em Painéis Elétricos onde, por critério de projeto, os conduítes são dimensionados com folgas.

No protótipo desenvolvido, este algoritmo foi implementado através de uma função de ordenação trivial que gera uma fila de interligações a partir da lista de interligações presente no modelo do painel e um iterador que chama as funções de permutação desta fila listadas no Algoritmo 4.1, codificado em Lua.

4.1.2 Inserção com *First Fail*

O algoritmo de Inserção com *First Fail* (IFF) é uma versão aperfeiçoada do algoritmo de Inserção Heurística Simples desenvolvida para tratar eficientemente alguns casos mais comuns de saturação de conduítes pela aplicação do conceito de *first fail*, isto é, a tentativa de antecipação das condições de falha durante a solução do problema.

O funcionamento do algoritmo é similar à Inserção Heurística Simples porém, caso a inserção de um cabo no grafo seja interrompida pela saturação de um ou mais conduítes, a interligação responsável pela falha é movida para o início da fila de inserção, uma propriedade que será mantida nas iterações futuras. Como, na primeira permutação subsequente, a interligação responsável pela falha é inserida em um grafo "vazio" (isto é, não modificado pela inserção de nenhum outro cabo), a ocorrência de uma nova falha nesta mesma interligação é a indicação clara de um modelo intratável, possivelmente em função de uma falha de projeto. Neste caso a execução do algoritmo é finalizada com a indicação do erro.

As características deste algoritmo são semelhantes às do algoritmo de Inserção Heurística Simples, com a adição de duas vantagens: (1) a possibilidade de recuperação de falhas triviais em tempo polinomial e (2) a capacidade de identificar alguns modelos intratáveis imediatamente após a ocorrência de uma falha. A aplicação de um algoritmo com estas propriedades é justificada pelas características das instâncias mais comuns do Problema do Roteamento de Cabos em Painéis Elétricos encontradas em aplicações industriais, onde apenas alguns trechos de conduítes são sujeitos a saturação.

A implementação deste algoritmo no protótipo é feita através de um "permutador com rolagem", listado no Algoritmo 4.2, que integra-se ao iterador apresentado no Algoritmo 4.1. O tratamento das falhas é feito através de uma chamada ao método **roll**, que receberá o índice da interligação responsável.

4.1.3 Inserção por Algoritmos Genéticos

O algoritmo de Inserção por Algoritmos Genéticos (IAG) foi desenvolvido para tratar instâncias específicas do Problema do Roteamento de Cabos em Painéis Elétricos onde exige-se a otimização do custo da fiação em um painel com muitos conduítes saturados e de instâncias viáveis que não são eficientemente tratadas pelos algoritmos anteriores.

Algoritmo 4.2 Permutador com rolagem para a estratégia *first fail*

```
local function RollPermutator(list)
    local o = {
        list = list,
        permutations = permutations(list)
    }
    -- 'Rola' a lista de permutações, movendo o elemento indicado
    -- para a primeira posição e marca a flag de rolagem.
    o.roll = function(self, n)
        local t = self.list[n]
        table.remove(self.list, n)
        table.insert(self.list, 1, t)
        self.rolled = true
    end
    -- Cria um iterador para as permutações da lista
    o.next = function(self)
        return function()
            if self.rolled then
                -- Se a lista foi 'rolada' por uma chamada anterior
                -- ao método 'roll', limpa a flag de rolagem e
                -- retorna a lista atual.
                self.rolled = false
                return self.list
            else
                -- Retorna uma permutação convencional da lista.
                return self.permutations()
            end
        end
    end
    return o
end
```

A geração da fila de inserção na etapa de inserção de interligações é um problema de sequenciamento para o qual algoritmos genéticos são frequentemente empregados. Este problema foi abordado usando algumas convenções aplicadas ao Problema do Caixeiro Viajante: a codificação cromossômica da lista de interligações é uma lista semelhante à usada na representação por caminhos para o PCV, adota-se o Operador PMX (*Partially Mapped Crossover*, Seção 2.5.2.1) como operador de *crossover*, o *Swap-Mutate* (Seção 2.5.3.1) como operador de mutação e a Seleção por Truncamento (Seção 2.5.1.2) como operador de seleção.

A função de *fitness* do algoritmo genético é listada no Algoritmo 4.3. A fim de permitir o aproveitamento de trechos viáveis do cromossomo de soluções inviáveis, esta função foi projetada para lidar simultaneamente com soluções viáveis e inviáveis. Para isto, o *fitness* $f(S)$ do indivíduo é dado por

$$f(S) = \begin{cases} M - custo(S) & \text{Se a solução for viável} \\ 0 - i & \text{Se a solução for inviável} \end{cases} \tag{4.1}$$

onde $custo(S)$ é a função do custo da solução candidata S, M é um número real suficientemente grande (superior ao maior custo possível para a instância do problema) e i é o índice da primeira interligação que não pôde ser inserida no painel em função da saturação dos conduítes. A opção pela seleção por truncamento (Seção 2.5.1.2) como operador de seleção para o AG é outra consequência desta decisão: na seleção por truncamento, os indivíduos são classificados em função do seu *fitness* e escolhe-se determinada porcentagem dos melhores colocados como progenitores das gerações futuras. No contexto do PRCPE isto significa que as n melhores soluções, válidas ou não, serão adotadas para a próxima geração.

A escolha destes três operadores transforma o algoritmo resultante em uma aplicação particular do *Breeder Genetic Algorithm* não distribuído (Seção 2.5.4). No protótipo desenvolvido para os testes, o algoritmo genético foi implementado em uma biblioteca independente intitulada GALILEO (*Genetic Algorithm Library for Intelligent Electrical Optimization*, ver Anexo II), chamada pelo aplicativo de roteamento.

4.2 Algoritmos para a Etapa de Roteamento

A etapa de roteamento é responsável por definir um conjunto de cabos para uma dada interligação e definir sua rota no grafo do painel pelo menor caminho disponível. As interligações contendo apenas dois terminais (aqui denominadas interligações ponto-a-ponto) podem ser

Algoritmo 4.3 Função objetivo para solução com algoritmos genéticos

```
local function obj(chrom)
    local queue = { }
    -- Inicializa a fila de inserção com dados do cromossomo.
    for i, j in ipairs(chrom) do
        queue[i] = model.conns[j]
    end
    -- Roteia; 'cables' é 'nil' se não houver solução viável.
    local cables, failedndx, failedconn = route_conn_list(
        queue, model.conduits, model.terminals,
        model.nodes, model.cabletypes)
    if cables then
        -- Há uma solução viável, calcula o fitness em função do
        -- custo total da solução.
        return maxcost - get_solution_cost(model, cables), cables
    else
        -- Não há solução viável, calcula o fitness em função
        -- do número de interligações inseridas com sucesso.
        return -#model.conns + failedndx, nil
    end
end
```

roteadas trivialmente usando um algoritmo de caminho mínimo – o protótipo desenvolvido usa o Algoritmo de Dijkstra (DIJKSTRA, 1959) –, porém, para todos os demais casos há a necessidade de determinar a sequência de conexão dos terminais e o caminho mínimo entre cada par de terminais, um problema equivalente a um Problema de Roteamento de Veículos (PRV, Seção 2.3). As Seções 4.2.1, 4.2.2 e 4.2.3 descrevem três algoritmos adotados para lidar com estes casos.

4.2.1 Roteamento por Busca Exaustiva

O Roteamento por Busca Exaustiva (Ex) consiste em gerar todas as sequências de conexão únicas possíveis para uma dada interligação e determinar os caminhos mínimos entre cada par de terminais usando o Algoritmo de Dijkstra. Dada a equivalência, em termos elétricos, entre um dado caminho e seu inverso, este algoritmo precisa avaliar apenas a metade das permutações possíveis para a lista de terminais. Após a avaliação das permutações, seleciona-se o caminho mais curto.

Este algoritmo possui tempo de execução exponencial (complexidade $O\left(\frac{n!}{2}\right)$ para n terminais), logo, não é aplicável para interligações com mais de uma dezena de terminais. Entretanto, a capacidade do algoritmo de determinar a solução ótima para uma interligação torna-o interessante para instâncias menores.

4.2.2 Roteamento por Busca Gulosa

O Roteamento por Busca Gulosa usa a heurística gulosa do vizinho mais próximo para determinar uma rota aceitável, ainda que não necessariamente ótima, entre um conjunto de terminais em tempo linear. O funcionamento do algoritmo pode ser resumido a:

- Dada uma interligação, elaborar uma matriz de conectividade contendo todas as rotas mais curtas únicas entre todos os terminais envolvidos. Esta etapa exige $n^2/2$ execuções do Algoritmo de Dijkstra para os n terminais da interligação;
- Selecionar os terminais mais distantes entre si como os terminais inicial e final da interligação;
- Partindo do terminal inicial, escolher o terminal de destino imediatamente mais próximo. Repetir o processo até atingir o terminal final.

Embora este algoritmo seja incapaz de garantir a rota ótima para uma interligação, sua aplicação é justificada pelo tempo de execução reduzido unido à distribuição característica dos componentes em certas instâncias práticas do Problema do Roteamento de Cabos em Painéis Elétricos e às possibilidades de otimização da implementação.

A implementação deste algoritmo desenvolvida para o protótipo avaliado usa algumas técnicas de programação dinâmica e reaproveitamento de resultados anteriores – *memoization*, conforme (CORMEN et al., 2001) – para minimizar, quando possível, o tempo consumido em execuções do Algoritmo de Dijkstra.

4.2.3 Roteamento por Colônias de Formigas

A eficiência da Otimização por Colônias de Formigas (ACO) no tratamento do Problema do Caixeiro Viajante (ver Seção 2.4) torna esta técnica uma candidata natural para aplicação na etapa de roteamento do Problema do Roteamento de Cabos em Painéis Elétricos, especialmente quando deseja-se encontrar uma solução ótima para instâncias intratáveis pela Busca Exaustiva.

Para a aplicação desta técnica na determinação das melhores rotas para uma interligação, adotou-se o algoritmo $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*, cuja superioridade relativa no tratamento do PCV é indicada pela revisão da literatura disponível (Seção 2.4.4). A variante adotada usa o critério de deposição de feromônio pela melhor formiga de cada iteração e usa, como condições

de parada, a obtenção de um número limite de iterações ou a estagnação da melhor solução por um determinado número de iterações.

A primeira etapa da aplicação do $\mathcal{MMAS}$ a uma interligação consiste em converter o problema da conexão de terminais para uma instância equivalente do Problema do Caixeiro Viajante. Isto é feito através da elaboração de uma matriz de rotas mínimas semelhante à usada no algoritmo de Busca Gulosa e a aplicação do algoritmo para a determinação de um caminho *fechado* entre todos os terminais envolvidos, isto é, a interligação entre os terminais na forma de um ciclo sem terminais inicial e final. Para a determinação destes terminais, abre-se o ciclo gerado de forma tal que os terminais mais distantes entre si tornam-se os terminais inicial e final, garantido, consequentemente, uma rota de custo mínimo.

Uma implementação do $\mathcal{MMAS}$ foi escrita especialmente para o protótipo desenvolvido, de modo a permitir algumas otimizações específicas para esta aplicação (especialmente no tratamento das matrizes de conectividade do grafo do painel). Algumas técnicas de *memoization* aplicadas ao roteamento por Busca Gulosa também são aplicadas à implementação deste algoritmo.

4.3 Implementação do Protótipo

Alguns critérios precisam ser considerados para a implementação dos algoritmos propostos em uma plataforma computacional, a exemplo da definição dos formatos de dados de entrada e das ferramentas de tratamento dos modelos dos painéis. As Seções 4.3.1, 4.3.2, 4.3.3 e 4.3.4 descrevem estas considerações e as ferramentas de desenvolvimento adotados para a programação de um protótipo aplicando os algoritmos descritos nas seções anteriores.

4.3.1 A Linguagem de Programação Lua

Os algoritmos de roteamento descritos foram implementados em Lua (IERUSALIMSCHY; FIGUEIREDO; CELES, 1996). Esta linguagem foi escolhida em função de um compromisso entre a agilidade de programação, velocidade de execução e consumo de recursos.

Lua é uma linguagem de programação interpretada a partir de *bytecode* em uma máquina virtual baseada em registradores e largamente aplicada como linguagem de extensão e *scripting*. A linguagem possui tipagem dinâmica, gerenciamento automático de memória por coletor de

lixo (*garbage collector*) e recursos particularmente úteis para descrição de dados e algoritmos (IERUSALIMSCHY; FIGUEIREDO; CELES, 1996). Há também um compilador *just-in-time* disponível para a plataforma x86 (PALL, 2009).

4.3.2 Linguagem de Descrição dos Modelos

A linguagem Lua também é usada para descrição das instâncias do problema, o que permite usar os recursos procedurais disponíveis para modelar os painéis, listas de cabos e bibliotecas de componentes. Esta característica é especialmente vantajosa para componentes complexos cuja definição, por meio de uma linguagem de configuração mais simples, seria excessivamente trabalhosa para o usuário.

O uso de uma linguagem de programação computacionalmente completa para descrição de dados cuja origem não é, necessariamente, confiável para o usuário traz riscos de segurança, como a execução de código nocivo. Para mitigar estes riscos, o protótipo implementado executa os arquivos de dados em uma *sandbox* que impede acesso ao ambiente global do interpretador Lua, oferecendo apenas as construções básicas da linguagem e algumas funções úteis para a descrição dos modelos.

Um exemplo da sintaxe usada nos arquivos de modelos de painéis é exibido no Algoritmo 4.4. Cada modelo deve retornar uma tabela com os campos **nodes**, relacionando o número de um nó do grafo de conduítes à sua posição no espaço do painel, **conduits**, listando os dados dos conduítes do painel (nós iniciais e finais, a área da seção transversal e o tipo de conduíte (aberto ou fechado), **terminals**, relacionando o nome de um terminal de ligação à sua posição no espaço do painel, **cabletypes**, relacionando o número de um tipo de cabo aos seus dados (descrição, área da seção transversal externa e custo) e **conns**, listando os dados das conexões do painel (lista de terminais conectados e tipo de cabo usado para a conexão).

Os modelos de painéis reais são mais complexos que o exemplo apresentado e sua descrição é notavelmente mais trabalhosa. Duas possíveis soluções para este problema são:

- Usar as construções procedurais da linguagem para modularizar a descrição do painel através de funções, iterações e chamadas a bibliotecas de dados externas, como demonstrado no modelo do painel PA1 (descrito na Seção 5.2.1 e listado no Anexo I);
- Caso o painel possua um projeto em CAD, gerar o arquivo do modelo automaticamente a

Algoritmo 4.4 Exemplo mínimo de um modelo de painel

```
return {
    nodes = {
        [1] = {   0,   0, 0 },
        [2] = { 400,   0, 0 },
        [3] = {   0, 100, 0 },
        [4] = { 400, 100, 0 },
        [5] = {   0, 200, 0 },
        [6] = { 400, 200, 0 }
    },
    conduits = {
        { f = 1, t = 2, s = 2500, o = true },
        { f = 3, t = 4, s = 2500, o = true },
        { f = 5, t = 6, s = 2500, o = true },
        { f = 1, t = 3, s = 2500, o = false },
        { f = 3, t = 5, s = 2500, o = false },
        { f = 2, t = 4, s = 2500, o = false },
        { f = 4, t = 6, s = 2500, o = false }
    },
    terminals = {
        ["Q1:1"] = { 30,  30, 20 },
        ["Q1:2"] = { 30,  70, 20 },
        ["K1:A1"] = { 30, 120, 60 },
        ["K1:A2"] = { 30, 180, 60 },
        ["P1:0"] = { 290, 30, 10 },
        ["P1:1"] = { 300, 30, 10 },
        ["P1:2"] = { 330, 30, 10 },
        ["P1:3"] = { 330, 80, 10 }
    },
    cabletypes = {
        [1] = { name = "1,0 CZ", es = 2, cost = 0.5 }
    },
    conns = {
        { terms = { "Q1:1", "K1:A1" }, ctype = 1 },
        { terms = { "P1:1", "P1:2", "P1:0" }, ctype = 1 }
    }
}
```

partir dos dados extraídos do projeto por um *plug-in* especialmente desenvolvido para o sistema em questão. Esta técnica foi usada para gerar o arquivo de dados do painel PA3 (ver Seção 5.2.3).

4.3.3 Visualização dos Modelos dos Painéis

Os resultados do roteamento dependem da corretude do modelo do painel, cuja verificação é relativamente complexa e trabalhosa. Para facilitar esta atividade, o protótipo inclui um recurso para visualização do arranjo espacial dos conduítes, nós e terminais (Figura 4.1). O visualizador também foi desenvolvido em Lua e usa uma biblioteca de *bindings* para acesso a funções da API gráfica OpenGL. A interface de usuário possui recursos para controle da exibição de conduítes, terminais e nós e operações de *zoom*, translação e rotação.

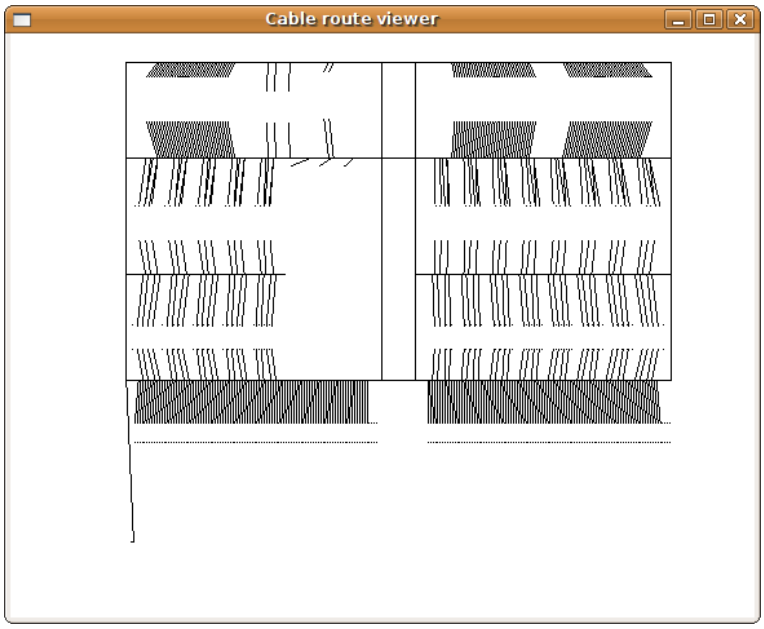


Figura 4.1: Visualizador de Modelos de Painéis

4.3.4 Funções e Recursos Auxiliares

Os algoritmos genéticos e de otimização por colônias de formigas usam valores aleatórios para a composição de populações e seleção, que podem ser marginalmente afetados pela randomicidade da sua função geradora (MAUCHER, 2009). A fim de minimizar esta distorção, o protótipo desenvolvido usa uma implementação do algoritmo *Mersenne twister*, com periodicidade de $2^{19937} - 1$

(MATSUMOTO; NISHIMURA, 1998), fornecida pela biblioteca "lrandom" (FIGUEIREDO, 2009). O gerador de números pseudoaleatórios da biblioteca padrão C, disponível nativamente no interpretador Lua, só é usado caso a "lrandom" não esteja disponível.

Alguns testes, como os listados na Seção 5.1, exigem medições do tempo de execução de trechos específicos do código do protótipo. A função `os.time()` da biblioteca padrão Lua é usada para totalizar o tempo de CPU empregado no intervalo de código selecionado, uma abordagem que desconsidera o tempo consumido em outros processos em execução no sistema, minimizando eventuais distorções[1].

4.4 Resumo do Capítulo

O Problema do Roteamento de Cabos em Painéis Elétricos pode ser dividido em duas etapas distintas para fins de implementação computacional: a etapa de inserção de interligações e a etapa de roteamento. Esta divisão permite a combinação de algoritmos distintos para cada tarefa.

A etapa de inserção de interligações pode ser tratada através de um algoritmo de Inserção Heurística Simples (IHS), uma estratégia geradora de permutações por *backtracking*, e uma versão aperfeiçoada para aumentar a eficiência na recuperação de falhas de inserção pela aplicação da estratégia *first fail*, denominada Inserção com *First Fail*. Algoritmos Genéticos podem ser aplicados a esta etapa, de forma a permitir o tratamento de instâncias não-triviais e fortemente saturadas.

A etapa de roteamento de interligações consiste em uma variante do Problema de Roteamento de Veículos. Pode-se tratar instâncias reduzidas com uma busca exaustiva que, porém, não pode ser aplicada à instâncias maiores devido a sua ordem de complexidade exponencial. Para tais casos, pode-se aplicar uma busca gulosa com ordem de complexidade linear, porém, esta estratégia é incapaz de garantir uma solução ótima. Como uma estratégia ponderada entre estes dois extremos, pode-se em aplicar um algoritmo de Otimização por Colônias de Formigas.

Um protótipo foi desenvolvido para aplicar estes algoritmos em instâncias práticas. Os testes com este aplicativo são descritos no próximo capítulo.

[1] A medição precisa do tempo de execução de processos em um sistema operacional multitarefa é particularmente complexa. Uma descrição detalhada das dificuldades e medidas corretivas é encontrada no cap. 9 de (BRYANT; O'HALLARON, 2002).

5 Testes e Resultados

Este capítulo apresenta os testes realizados com a solução proposta no capítulo anterior usando instâncias do problema baseadas em modelos artificiais (Seção 5.1) e práticos (Seção 5.2). A avaliação dos resultados obtidos é apresentada na Seção 5.3.

Todos os testes descritos foram executados em um computador pessoal com processador Intel Core 2 Duo T7100, com dois núcleos de processamento a 1,80 GHz, 2 GiB[1] de memória RAM e 2 MiB de memória *cache*, executando um sistema operacional Linux com *kernel* 2.6.31 executando em modo 32 bits. Todos os testes foram executados de forma controlada de modo a minimizar a influência de fatores externos sobre os resultados obtidos.

5.1 Testes com Modelos Artificiais

Os testes com modelos artificiais avaliam características conhecidas dos algoritmos de inserção de cabos e roteamento usando instâncias mínimas necessárias para representá-las de forma isolada. Estes modelos não representam painéis reais, mas podem ser encontrados como subproblemas de muitos painéis típicos. As Seções 5.1.1, 5.1.2 e 5.1.3 descrevem testes com três modelos desta classe.

5.1.1 Modelo PB1

O teste PB1 objetiva avaliar o comportamento dos algoritmos de roteamento na minimização da sequência de conexão de interligações com mais de dois terminais, uma ocorrência muito comum em painéis reais. Como este procedimento é uma especialização do Problema do Caixeiro Viajante, espera-se que a relação entre o número de terminais e o tempo de execução seja exponencial para a busca exaustiva e linear para o algoritmo guloso, com a otimização por colônia de formigas situando-se em uma posição intermediária.

O modelo desenvolvido para este teste (Figura 5.1) é composto por uma única régua X1 com

[1] Neste documento, usa-se os prefixos binários definidos pelas normas IEC 60027-2 (2000), IEEE 1541-2002 e ISO/IEC 80000 (2008) para desambiguar grandezas múltiplas de 1000 e 1024, tal que 1 KiB equivale a 1024 B ou 1,024 KB.

20 bornes e uma canaleta, mapeada como um conduíte do tipo aberto, paralela à régua. Para evitar a influência de outros fatores, os únicos terminais existentes no modelo são os bornes de X1, há apenas um tipo de cabo e a seção transversal da canaleta é suficientemente grande para evitar a saturação.

O circuito elétrico do modelo possui 18 interligações, cada uma possuindo entre três a 20 terminais. A primeira interligação conecta os bornes [1, 2, 3] da régua X1, a segunda interligação conecta os bornes [1, 2, 3, 4], e assim sucessivamente até a 18ª interligação conectando os bornes [1, 2, . . . , 19, 20]. A participação do mesmo terminal em mais de uma interligação não é comum em projetos práticos (exceto quando este terminal é uma barramento), porém, esta situação não é restrita pelos algoritmos de roteamento e não afeta os resultados.

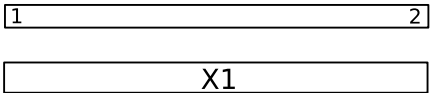


Figura 5.1: Modelo PB1

O procedimento de teste consiste em medir o tempo de execução da etapa de roteamento (desconsiderando o tempo empregado para a preparação do interpretador Lua, carregamento dos dados, seccionamento de conduítes, etc.) para cada uma das interligações usando os algoritmos de busca exaustiva, busca gulosa e otimização por colônia de formigas. Para minimizar as distorções provocadas por eventuais imprecisões nas medidas, considera-se a média dos tempos medidos em cinco execuções de cada algoritmo.

O gráfico da Figura 5.2 apresenta os resultados obtidos, com o número de terminais representado no eixo horizontal, o tempo de execução no eixo vertical e os algoritmos busca exaustiva (Ex), busca gulosa (BG) e otimização por colônia de formigas $\mathcal{MMAS}$ (ACO) representados, respectivamente, pelas curvas tracejada, contínua e pontilhada. Os tempos de execução inferiores à granularidade das medições são indicados como zero e o teste com a busca exaustiva foi limitado a oito terminais. O algoritmo de otimização por colônia de formigas foi configurado para usar uma população de duas formigas por terminal na lista de interligação, peso da quantidade de feromônio $\alpha = 1$, peso da função heurística $\beta = 2$, fator de evaporação do feromônio $\tau_{ev} = 0{,}05$, taxa de incremento da trilha de feromônio $\tau_{incr} = 5/l$ (onde l é o comprimento total da rota), concentração de feromônio limitada entre o mínimo $\tau_{min} = 0{,}01$ e máximo $\tau_{max} = 2$ e, como critérios de parada, adotou-se o limite máximo de 100 iterações ou estagnação da melhor solução por 20 iterações.

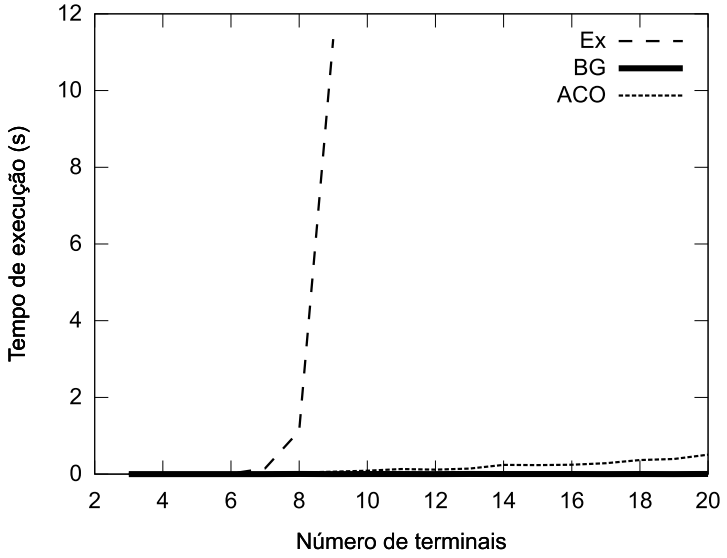


Figura 5.2: Tempos de execução para o Modelo PB1

5.1.2 Modelo PB2

O teste PB2 objetiva avaliar o comportamento dos algoritmos de inserção de interligações na detecção de soluções inviáveis e tratamento da saturação dos conduítes. Estas situações são comumente encontradas em painéis reais com conduítes subdimensionados, que demandam a busca de rotas alternativas para permitir a execução de algumas interligações. Dadas as características do problema, espera-se que o algoritmo de inserção com heurística simples demore a encontrar uma solução viável, o algoritmo de inserção heurística com *first fail* encontre-a rapidamente e o algoritmo genético encontre-a em um tempo intermediário.

O modelo desenvolvido para este teste é exibido na Figura 5.3. O projeto possui quatro componentes (A1, A2, A3 e A4), com terminais de ligação localizados na parte superior, dispostos entre as canaletas. O circuito elétrico possui 10 interligações: cinco entre terminais dos componentes A2 e A3 usando cabos com custo de R$0,50/m e cinco entre terminais dos componentes A1 e A4 usando um cabo com custo de R$1,00/m e as canaletas foram dimensionadas para permitir a passagem de apenas seis cabos em qualquer segmento.

A configuração peculiar deste modelo não permite uma solução trivial, pois a passagem dos cabos mais caros interligando A1 e A4 pelo caminho mais curto (1–2) impedirá a montagem do painel, pois não haverá espaço suficiente nesta canaleta para todos os cabos que interligam os componentes A2 e A3 e que não admitem rotas alternativas. A solução ótima consiste em passar os cinco cabos interligando os componentes A2 e A3 e um dos cabos interligando A1 e A4 pela

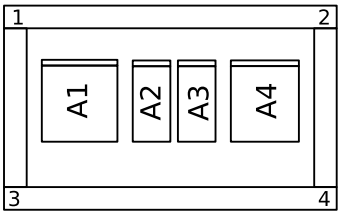


Figura 5.3: Modelo PB2

canaleta 1–2 e os outros quatro cabos entre A1 e A4 pelo caminho 1–3–4–2.

O procedimento de teste consiste em executar o roteamento usando os algoritmos de inserção heurística simples (HS), inserção heurística com *first fail* (FF) e inserção com algoritmo genético (AG) e comparar os resultados obtidos e o tempo de execução total. Como todas as interligações envolvem apenas dois terminais cada, não há necessidade de determinar a sequência de interligação, inexistindo qualquer influência dos algoritmos da etapa de roteamento.

Tabela 5.1: Parâmetros para o algoritmo genético

Parâmetro	Valor
Tamanho da população	20 indivíduos
Ponto de corte (c_t)	50%
Probabilidade de mutação (p_m)	0,05
Critérios de parada	Limite de 100 gerações Estagnação da melhor solução por 20 gerações

A Tabela 5.1 descreve os parâmetros de configuração do algoritmo genético para a execução dos testes. Os tempos foram medidos através do comando `time` do Linux e incluem o período necessário para carregar o interpretador Lua, o programa e o arquivo de dados, validar o modelo e processar o seccionamento dos conduítes abertos. Nas medições, considera-se apenas o tempo de processamento consumido pelo processo do interpretador, excluindo a influência de outros processos em execução no sistema. A fim de minimizar as distorções provocadas por imprecisões nas medidas dos tempos de execução e a influência dos valores aleatórios no algoritmo genético, todos os testes foram executados cinco vezes.

Os resultados obtidos são listados na Tabela 5.2. Os tempos e custos obtidos são listados para cada execução do algoritmo indicada na coluna “Execução” e a linha “Média” representa

as média dos tempos de execução e dos custos obtidos para todas as cinco execuções de cada algoritmo.

Tabela 5.2: Resultados dos testes do modelo PB2

Algoritmo	Execução	Custo (R$)	Tempo (s)
HS	1	4,60	56,884
	2	4,60	61,360
	3	4,60	60,856
	4	4,60	61,176
	5	4,60	61,588
	Média	4,60	60,373
FF	1	4,60	0,148
	2	4,60	0,148
	3	4,60	0,156
	4	4,60	0,148
	5	4,60	0,156
	Média	4,60	0,151
AG	1	4,60	0,144
	2	4,60	0,152
	3	4,60	0,148
	4	4,60	0,160
	5	4,60	0,140
	Média	4,60	0,149

5.1.3 Modelo PB3

O teste PB3 visa avaliar o comportamento dos algoritmos de inserção de interligações na minimização do custo de um painel com conduítes saturados, ou seja, o teste simula uma situação onde, perante a existência de vários conduítes saturados, precisa-se encontrar um comprometimento ideal entre a rota e o custo dos cabos a fim de minimizar o custo total da solução. Dadas as características do problema, espera-se que todos os algoritmos encontrem a solução ideal, variando apenas o tempo de execução.

O modelo desenvolvido para este teste é exibido na Figura 5.4. O projeto possui dois componentes (A1 e A2) dispostos entre nas extremidades de duas canaletas horizontais que são interligadas por oito canaletas verticais. O circuito elétrico possui 30 interligações conectando terminais dos componentes A1 e A2 usando seis tipos distintos de cabos com custos de R$1,50/m, R$1,25/m, R$1,00/m, R$0,75/m, R$0,50/m e R$0,25/m. A área da seção transversal das canaletas horizontais é suficiente para evitar a saturação, porém, cada canaleta vertical permite a passagem de apenas cinco cabos.

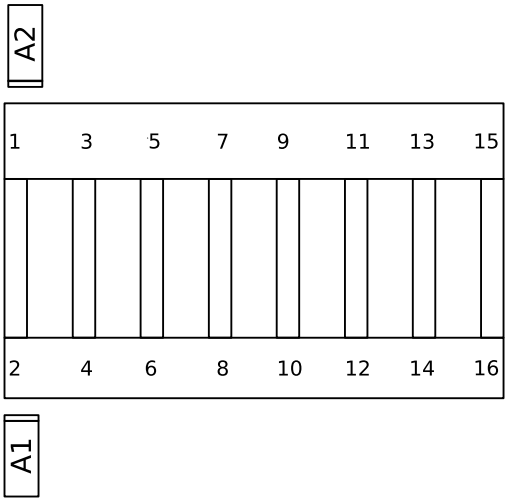


Figura 5.4: Modelo PB3

Esta configuração obriga os algoritmos a conduzir os cabos mais caros pela canaleta 1–2, obtendo as rotas mais curtas, e ocupar progressivamente as rotas mais longas com cabos mais baratos. As rotas mais longas devem ficar vazias. O procedimento de teste consiste em executar o roteamento usando os algoritmos de inserção heurística simples (HS), inserção heurística com *first fail* (FF) e inserção com algoritmo genético (AG) e comparar o custo e o tempo de execução dos resultados obtidos. Como todas as interligações envolvem apenas dois terminais cada, não há necessidade de determinar a sequência de interligação, inexistindo qualquer influência dos algoritmos da etapa de roteamento.

Os procedimentos de tomada de tempos e os parâmetros de configuração para o algoritmo genético são idênticos aos usados no teste do modelo PB2 (ver Tabela 5.1). Os resultados obtidos são listados na Tabela 5.3. Os tempos e custos obtidos são listados para cada execução do algoritmo indicada na coluna “Execução” e a linha “Média” representa as média dos tempos de execução e dos valores obtidos para todas as cinco execuções de cada algoritmo.

Tabela 5.3: Resultados dos testes do modelo PB3

Algoritmo	Execução	Custo (R$)	Tempo (s)
HS	1	12,84	0,020
	2	12,84	0,024
	3	12,84	0,020
	4	12,84	0,024
	5	12,84	0,024
	Média	12,84	0,022
FF	1	12,84	0,020
	2	12,84	0,016
	3	12,84	0,024
	4	12,84	0,020
	5	12,84	0,016
	Média	12,84	0,019
AG	1	13,29	12,917
	2	13,29	10,337
	3	13,74	7,128
	4	13,34	10,557
	5	13,29	11,501
	Média	13,39	10,488

5.2 Testes com Modelos Práticos

Os testes com modelos práticos permitem avaliar o comportamento dos algoritmos em projetos reais, sujeitos às condições normalmente encontradas na indústria. Três modelos de painéis foram desenvolvidos para estes testes, dois seguindo critérios de projeto tipicamente aplicados em painéis de automação industrial e um modelando um painel efetivamente construído e instalado. As Seções 5.2.1, 5.2.2 e 5.2.3 descrevem os testes com três modelos desta classe.

5.2.1 Painel PA1

O painel PA1, cujo *layout* mecânico pode ser visto na Figura 5.5, é o projeto de um acionamento de cinco motores por partidas diretas em uma configuração largamente encontrada em aplicações industriais. Este painel é conectado à rede elétrica através de três barras de entrada (A, B e C), aos motores através dos bornes da régua X1 e ao terra de proteção pela barra de aterramento PE. O circuito elétrico correspondente é apresentado na Figura 5.6.

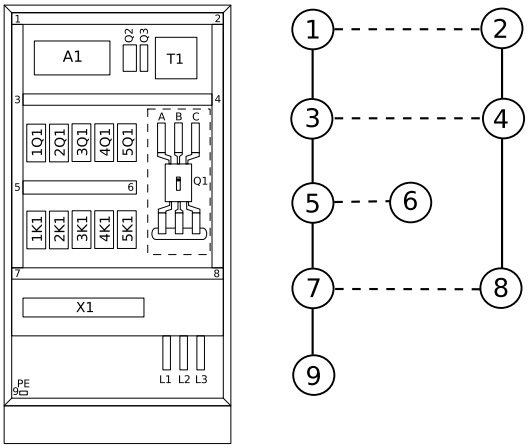


Figura 5.5: *Layout* mecânico e grafo de conduítes do painel PA1

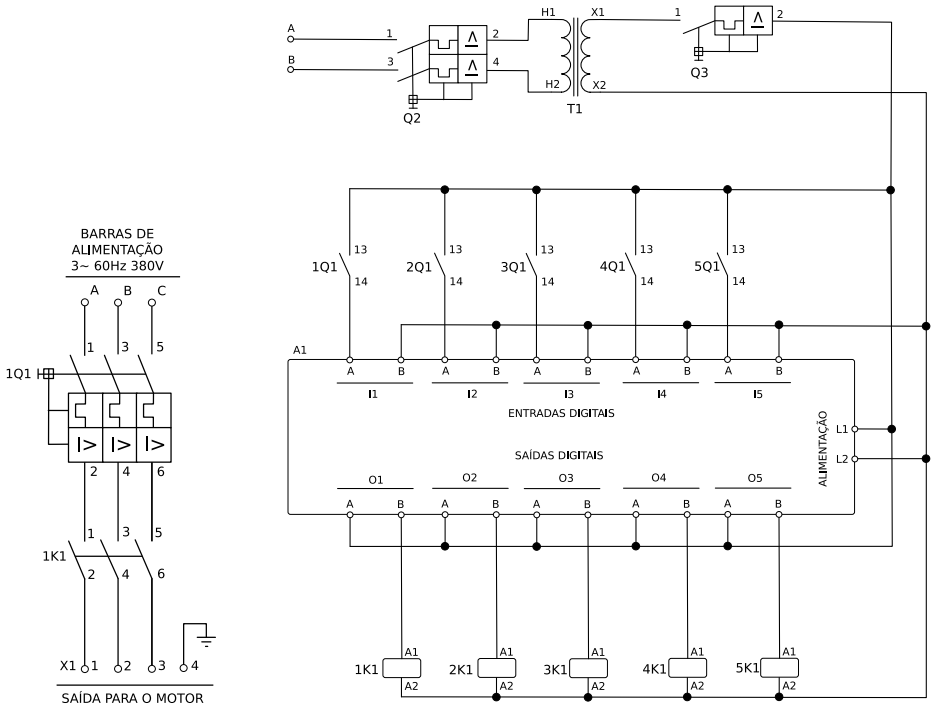


Figura 5.6: Uma partida direta e circuito de comando do painel PA1

O acionamento das partidas é feito por um controlador lógico programável (A1) composto por 16 entradas digitais isoladas, com terminais identificados como “I*n*A – I*n*B”, e 16 saídas

digitais a relê, com terminais identificados como "O*n*A – O*n*B". Cada partida direta é composta por um contator tripolar e um disjuntor termomagnético. Os contatos auxiliares dos disjuntores e as bobinas de comando dos contatores são ligados, respectivamente, às entradas e saídas do CLP. O circuito de comando é alimentado por um transformador monofásico (T1) protegido por disjuntores ligados aos enrolamentos primário (Q2) e secundário (Q3).

As interligações de cada partida direta do painel são listadas na Tabela 5.4 e a Tabela 5.5 lista as interligações do circuito de comando que não estão associadas a nenhuma partida específica. Nestas tabelas, os pontos de ligação "fase" e "neutro" representam pontos de mesmo potencial elétrico que devem ser conectados pelo aplicativo de roteamento da forma mais eficiente possível. Os dados dos cabos usados para estas interligações são listados na Tabela 5.6. O modelo completo possui 67 interligações que geram 87 cabos e é listado no Anexo I.

A fiação do painel será alojada em canaletas plásticas com seção transversal de 80x30 mm, conforme o *layout* e o grafo da Figura 5.5. As canaletas usadas neste painel são vazadas para permitir a passagem de cabos, porém, por questões de estética e organização dos cabos, apenas as canaletas representadas com linhas tracejadas no grafo de conduítes da Figura 5.5 são modeladas como conduítes do tipo "aberto". A aresta ligando os nós 7 e 9 não representa uma canaleta, mas uma área destinada a passagem de um chicote com os cabos ligados à barra de aterramento PE. Dado o espaço interno existente nas canaletas usadas e a organização dos componentes do painel, não espera-se a existência de conduítes saturados.

5.2.1.1 Teste dos Algoritmos de Roteamento

O teste dos algoritmos de roteamento consiste em executar operações de roteamento com cada um dos algoritmos propostos a fim de determinar seu desempenho para este modelo específico, considerando a qualidade da solução, sumarizada pelo custo total da cablagem do painel, e o tempo de execução.

A Tabela 5.7 apresenta os resultados das avaliações feitas com os algoritmos de inserção heurística simples (HS), inserção heurística com *first fail* (FF) e otimização com algoritmos genéticos (AG). A etapa de roteamento das interligações usa o algoritmo guloso (BG) e otimização por colônia de formigas com $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* (ACO). A busca exaustiva não foi avaliada, pois o tempo de execução esperado para as duas interligações com mais de dois terminais – "Fase" e "Neutro", com 12 terminais cada, implicando em uma complexidade computacional de $O(12!)$ – extrapolará o aceitável para uma aplicação prática. Em todos os

Tabela 5.4: Cabos de uma partida do painel PA1

Cor	Bitola (mm^2)	De	Para
Preto	6	Barra A	1Q1:1
Preto	6	Barra B	1Q1:3
Preto	6	Barra C	1Q1:5
Preto	2,5	1Q1:2	1K1:1
Preto	2,5	1Q1:4	1K1:3
Preto	2,5	1Q1:6	1K1:5
Preto	2,5	1K1:2	X1:1
Preto	2,5	1K1:4	X1:2
Preto	2,5	1K1:6	X1:3
Verde/Amarelo	2,5	Barra PE	X1:4
Cinza	1	Fase	1Q1:13
Cinza	1	1Q1:14	A1:I1A
Cinza	1	A1:I1B	Neutro
Cinza	1	Fase	A1:O1A
Cinza	1	A1:O1B	1K1:A1
Cinza	1	1K1:A2	Neutro

testes, usou-se a distância de seccionamento de 10 mm (testes com outros valores são listados na Seção 5.2.1.2).

Os parâmetros de configuração para o algoritmo genético são idênticos aos usados no teste do modelo PB2 (ver Tabela 5.1, página 75), obtidos empiricamente em testes com outros modelos. O algoritmo de otimização por colônia de formigas foi configurado para usar uma população de duas formigas por terminal na lista de interligação, peso da quantidade de feromônio $\alpha = 1$, peso da função heurística $\beta = 2$, fator de evaporação do feromônio $\tau_{ev} = 0{,}05$, taxa de incremento da trilha de feromônio $\tau_{incr} = 5/l$ (onde l é o comprimento total da rota), concentração de feromônio limitada entre o mínimo $\tau_{min} = 0{,}01$ e máximo $\tau_{max} = 2$ e, como critérios de parada,

Tabela 5.5: Cabos de alimentação do circuito de comando do painel PA1

Cor	Bitola (mm^2)	De	Para
Preto	2,5	Barra A	Q2:1
Preto	2,5	Barra B	Q2:3
Preto	2,5	Q2:2	T1:H1
Preto	2,5	Q2:4	T1:H2
Cinza	1	T1:X1	Q3:1
Cinza	1	Q3:2	Fase
Cinza	1	T1:X2	Neutro

Tabela 5.6: Dados dos cabos usados para os painéis PA1 e PA2

Cor	Bitola (mm^2)	Seção transversal externa (mm^2)	Custo (R$/m)
Cinza	1	2	0,25
Preto	2,5	6	0,50
Preto	6	12	0,85
Verde/Amarelo	2,5	6	0,50

adotou-se o limite máximo de 100 iterações ou estagnação da melhor solução por 20 iterações.

Um exemplo de saída do algoritmo de roteamento pode ser visto na Tabela 5.8. Esta tabela lista o comprimento dos cabos gerados para cada interligação especificada, seus pontos de origem e destino e a rota percorrida tomando como referência os números dos nós indicados na Figura 5.5. O total de cabos usados para esta solução é apresentado na Tabela 5.9 e as demais tabelas de resultados são omitidas para brevidade.

Os tempos de execução dos algoritmos e o custo total das soluções obtidas são listados na Tabela 5.7. A fim de minimizar as distorções provocadas por imprecisões nas medidas, os valores exibidos são os valores médios obtidos a partir de cinco execuções de cada algoritmo. Os tempos foram medidos com o mesmo procedimento descrito para o teste do modelo PB2 (página 74).

Tabela 5.7: Custos totais e tempos médios de execução dos algoritmos para o modelo PA1

Algoritmo	Custo (R$)	Tempo (s)
HS + BG	17,94	1,685
HS + ACO	18,31	2,011
FF + BG	17,94	1,691
FF + ACO	18,30	2,036
AG + BG	17,94	734,702
AG + ACO	18,22	1183,958

Tabela 5.8: Resultados de um teste de roteamento com o modelo PA1 com seccionamento de conduítes em 10 mm e heurísticas FF+BG

De	Rota	Para	Cabo	Comprimento (m)
A		2Q1:1	6,0 PT	0,64
A		5Q1:1	6,0 PT	0,46
B		5Q1:3	6,0 PT	0,5
C		5Q1:5	6,0 PT	0,53
B		4Q1:3	6,0 PT	0,56
B		2Q1:3	6,0 PT	0,68
C		4Q1:5	6,0 PT	0,59
C		2Q1:5	6,0 PT	0,71
A		4Q1:1	6,0 PT	0,52
B		3Q1:3	6,0 PT	0,62
A		3Q1:1	6,0 PT	0,58
C		3Q1:5	6,0 PT	0,65
A		1Q1:1	6,0 PT	0,7
C		1Q1:5	6,0 PT	0,77

De	Rota	Para	Cabo	Comprimento (m)
B		1Q1:3	6,0 PT	0,74
3K1:4		X1:10A	2,5 PT	0,26
4Q1:4		4K1:3	2,5 PT	0,19
PE	9 7	X1:12A	2,5 VD/AM	0,51
3K1:6		X1:11A	2,5 PT	0,27
4Q1:2		4K1:1	2,5 PT	0,19
4Q1:6		4K1:5	2,5 PT	0,19
A	4 2	Q2:1	2,5 PT	0,92
3K1:2		X1:9A	2,5 PT	0,26
5K1:4		X1:18A	2,5 PT	0,34
5K1:2		X1:17A	2,5 PT	0,34
5K1:6		X1:19A	2,5 PT	0,35
PE	9 7	X1:20A	2,5 VD/AM	0,55
4K1:2		X1:13A	2,5 PT	0,3
5Q1:6		5K1:5	2,5 PT	0,19
5Q1:2		5K1:1	2,5 PT	0,19
4K1:4		X1:14A	2,5 PT	0,3
5Q1:4		5K1:3	2,5 PT	0,19
4K1:6		X1:15A	2,5 PT	0,31
PE	9 7	X1:16A	2,5 VD/AM	0,53
3Q1:6		3K1:5	2,5 PT	0,19
3Q1:4		3K1:3	2,5 PT	0,19
3Q1:2		3K1:1	2,5 PT	0,19
PE	9 7	X1:4A	2,5 VD/AM	0,47
T1:X1	4 2	Q3:1	2,5 PT	0,69

De	Rota	Para	Cabo	Comprimento (m)
Q2:4	4 2	T1:H2	2,5 PT	0,71
2Q1:2		2K1:1	2,5 PT	0,19
1Q1:2		1K1:1	2,5 PT	0,19
1K1:6		X1:3A	2,5 PT	0,19
1Q1:4		1K1:3	2,5 PT	0,19
1K1:4		X1:2A	2,5 PT	0,18
1K1:2		X1:1A	2,5 PT	0,18
2Q1:4		2K1:3	2,5 PT	0,19
2K1:2		X1:5A	2,5 PT	0,22
2Q1:6		2K1:5	2,5 PT	0,19
B	4 2	Q2:3	2,5 PT	0,85
PE	9 7	X1:8A	2,5 VD/AM	0,49
Q2:2	4 2	T1:H1	2,5 PT	0,74
2K1:4		X1:6A	2,5 PT	0,22
2K1:6		X1:7A	2,5 PT	0,23
1Q1:6		1K1:5	2,5 PT	0,19
5Q1:14	3 1	A1:I5A	1,0 CZ	0,83
A1:O5B	3 5	5K1:A1	1,0 CZ	0,86
4Q1:14	3 1	A1:I4A	1,0 CZ	0,76
A1:L1	1 3	A1:O1A	1,0 CZ	0,5
A1:O1A		A1:O2A	1,0 CZ	0,18
A1:O2A		A1:O3A	1,0 CZ	0,18
A1:O3A		A1:O4A	1,0 CZ	0,18
A1:O4A		A1:O5A	1,0 CZ	0,18
A1:O5A		2Q1:13	1,0 CZ	0,22

De	Rota	Para	Cabo	Comprimento (m)
2Q1:13		1Q1:13	1,0 CZ	0,3
1Q1:13		3Q1:13	1,0 CZ	0,36
3Q1:13		4Q1:13	1,0 CZ	0,3
4Q1:13		Q3:2	1,0 CZ	0,29
Q3:2		5Q1:13	1,0 CZ	0,23
2Q1:14	3 1	A1:I2A	1,0 CZ	0,62
A1:O2B	3 5	2K1:A1	1,0 CZ	0,65
A1:O4B	3 5	4K1:A1	1,0 CZ	0,79
3Q1:14	3 1	A1:I3A	1,0 CZ	0,69
A1:O1B	3 5	1K1:A1	1,0 CZ	0,58
A1:O3B	3 5	3K1:A1	1,0 CZ	0,72
1Q1:14	3 1	A1:I1A	1,0 CZ	0,55
A1:I5B		A1:I4B	1,0 CZ	0,18
A1:I4B		A1:I3B	1,0 CZ	0,18
A1:I3B		A1:I2B	1,0 CZ	0,18
A1:I2B		A1:I1B	1,0 CZ	0,18
A1:I1B	1 3	A1:L2	1,0 CZ	0,5
A1:L2		T1:X2	1,0 CZ	0,57
T1:X2	4 8	5K1:A2	1,0 CZ	0,97
5K1:A2		4K1:A2	1,0 CZ	0,21
4K1:A2		3K1:A2	1,0 CZ	0,21
3K1:A2		2K1:A2	1,0 CZ	0,21
2K1:A2		1K1:A2	1,0 CZ	0,21

Tabela 5.9: Totais dos cabos apresentados na Tabela 5.8

Bitola (mm^2)	Cor	Comprimento (m)	Custo (R$)
1,0	Cinza	13,58	3,39
2,5	Preto	10,73	5,36
6,0	Preto	9,31	7,91
2,5	Verde/Amarelo	2,54	1,27

5.2.1.2 Teste das Distâncias de Seccionamento de Conduítes

O teste das distâncias de seccionamento de conduítes consiste em aplicar repetidamente um algoritmo de roteamento com valores distintos para a distância de seccionamento de conduítes a fim de quantificar a distorção provocada sobre os resultados obtidos e, consequentemente, avaliar sua influência perante a qualidade dos resultados obtidos.

Os resultados obtidos são exibidos na Tabela 5.10. O roteamento foi executado usando o algoritmo de inserção heurística com *first fail* e roteamento guloso para o vizinho mais próximo (FF+BG), para cada valor de seccionamento indicado na coluna "Seccionamento", resultando em um modelo derivado com o número de conduítes e nós, gerados pelo seccionamento, indicados nas respectivas colunas. O tempo de execução indicado é a média aritmética dos valores medidos em cinco execuções do aplicativo para cada valor de seccionamento. Este procedimento destina-se exclusivamente a reduzir eventuais imprecisões na tomada dos tempos, pois, como os algoritmos usados neste teste são determinísticos, não há diferenças nos resultados entre as execuções.

Distâncias de seccionamento inferiores a 10 mm possuem pouca aplicação prática para um painel como este, dada a distância típica entre os terminais e a pequena influência exercida sobre o comprimento dos cabos. A quantidade de conduítes gerados no modelo derivado, entretanto, torna estes valores úteis como um recurso para avaliação de desempenho do algoritmo.

A distorção dos resultados, provocada por conduítes excessivamente longos, reflete-se no custo total de cada solução como consequência da determinação imprecisa do comprimento dos cabos. Este efeito pode ser visto na Figura 5.7, que representa o caminho dos cabos no painel para as distâncias de seccionamento de 250, 100, 50, 20, 10 e 5 mm.

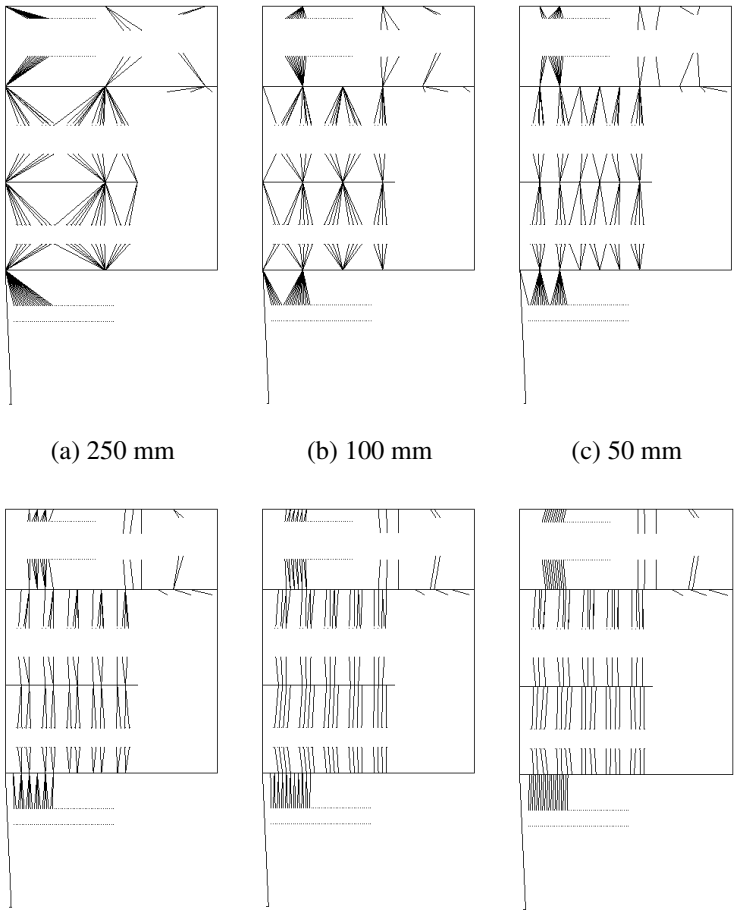
Tabela 5.10: Influência do seccionamento de conduítes para o modelo PA1

Seccionamento	Custo total (R$)	Conduítes	Nós	Tempo (s)
500	28,06	13	12	0,040
250	20,73	17	16	0,047
100	18,29	28	27	0,077
75	18,04	35	34	0,090
50	18,07	46	45	0,126
30	17,97	72	71	0,237
20	17,82	104	103	0,498
10	17,94	202	201	1,710
5	17,88	394	393	7,570
1	17,88	1930	1929	181,009

5.2.2 Painel PA2

O painel PA2 (Figura 5.8) é uma ampliação do painel PA1, desenvolvida para avaliar o comportamento dos algoritmos de roteamento em um conjunto de painéis interligados. O circuito das partidas diretas (Figura 5.6) foi mantido, porém seu número foi elevado para 13 e dois novos *racks* de entradas e saídas para o CLP (A2 e A3), com 15 entradas e 15 saídas digitais cada, foram adicionados. As entradas e saídas não usadas para o comando das partidas diretas estão disponíveis para uso em campo, ligadas aos bornes da régua X2. Os componentes adicionais são acondicionados em uma nova coluna, acoplada diretamente à coluna já existente. O modelo completo do circuito elétrico deste painel possui 303 interligações que geram 359 cabos.

A Figura 5.9 apresenta o grafo de conduítes do painel. Como não há nenhuma barreira mecânica entre as duas colunas, a passagem de cabos é feita através de recortes nas paredes das canaletas, modelados como os conduítes 2-10, 4-12 e 8-16. Os mesmos tipos de cabos usados para o Modelo PA1 são usados para este painel (Tabela 5.6).



(a) 250 mm (b) 100 mm (c) 50 mm

(d) 20 mm (e) 10 mm (f) 5 mm

Figura 5.7: Distorção das rotas do modelo PA1 para diversas distâncias de seccionamento

5.2.2.1 Teste dos Algoritmos de Roteamento

A Tabela 5.11 apresenta os resultados das avaliações feitas com os algoritmos de inserção heurística simples (HS), inserção heurística com *first fail* (FF) e inserção por algoritmos genéticos (AG). A etapa de roteamento das interligações usa o algoritmo guloso (BG) e otimização por colônia de formigas com $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* (ACO). A fim de minimizar as distorções provocadas por imprecisões nas medidas, os valores exibidos são os valores médios obtidos a partir de cinco execuções de cada algoritmo. A busca exaustiva não foi aplicada, pois as duas interligações com mais de dois terminais (“Fase” e “Neutro”, com 30 terminais cada) extrapolam o limite de tempo aceitável para este algoritmo. Os testes com algoritmos genéticos foram cancelados quando o tempo de execução superou 90 minutos. Em todos os testes, usou-se a distância de seccionamento de 10 mm (testes com outros valores são listados na Seção 5.2.2.2).

O algoritmo genético foi configurado com os mesmos parâmetros usados nos testes do

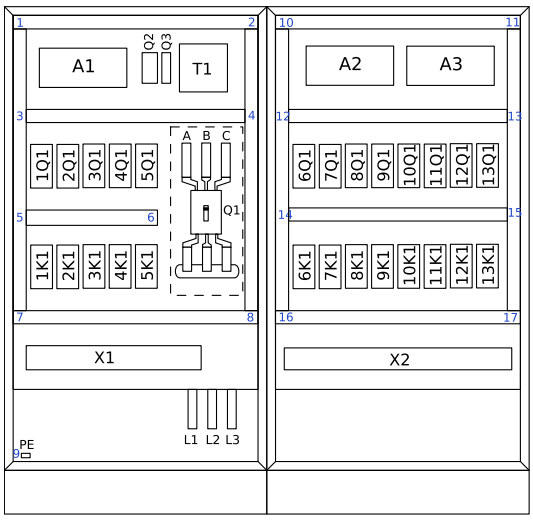


Figura 5.8: *Layout* mecânico do painel PA2

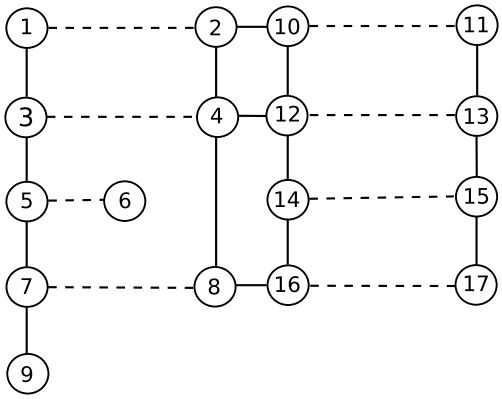


Figura 5.9: Grafo de conduítes do painel PA2

modelo PB2 (ver Tabela 5.1, página 75) e o algoritmo de otimização por colônia de formigas foi configurado para usar uma população de duas formigas por terminal na lista de interligação, peso da quantidade de feromônio $\alpha = 1$, peso da função heurística $\beta = 2$, fator de evaporação do feromônio $\tau_{ev} = 0{,}05$, taxa de incremento da trilha de feromônio $\tau_{incr} = 5/l$ (onde l é o comprimento total da rota), concentração de feromônio limitada entre o mínimo $\tau_{min} = 0{,}01$ e máximo $\tau_{max} = 2$ e, como critérios de parada, adotou-se o limite máximo de 100 iterações ou estagnação da melhor solução por 20 iterações.

Tabela 5.11: Custos totais e tempos médios de execução dos algoritmos para o modelo PA2

Algoritmo	Custo (R$)	Tempo (s)
HS + BG	93,13	30,93
HS + ACO	97,02	34,65
FF + BG	93,13	31,52
FF + ACO	96,81	36,78
AG + BG	–	>5400
AG + ACO	–	>5400

5.2.2.2 Teste das Distâncias de Seccionamento de Conduítes

A Tabela 5.12 apresenta os resultados obtidos com diversos valores de seccionamento de conduítes. O roteamento foi executado usando o algoritmo de inserção heurística com *first fail* e roteamento guloso para o vizinho mais próximo (FF+BG), para cada valor de seccionamento indicado na coluna "Seccionamento", resultando em um modelo derivado com o número de conduítes e nós, gerados pelo seccionamento, indicados nas respectivas colunas. O tempo de execução indicado é a média aritmética dos valores medidos em cinco execuções do aplicativo para cada valor de seccionamento. Este procedimento destina-se exclusivamente a reduzir eventuais imprecisões na tomada dos tempos, pois, como os algoritmos usados neste teste são determinísticos, não há diferenças nos resultados entre as execuções.

5.2.3 Painel PA3

O Painel PA3 é um projeto obtido de uma aplicação real[2] usado para avaliar a eficiência do seccionamento de conduítes e do roteamento ponto a ponto, comparados aos dados fornecidos por um especialista humano, em uma coluna com configuração mecânica relativamente complexa. Trata-se de um painel de remotas de um CLP Bosch CL200 composto por dois *racks* tipo GG3 (BOSCH, 2001) configurados com um cartão de rede PROFIBUS e cartões de E/S digitais e analógicas. Todas as entradas e saídas são ligadas a bornes para uso em campo, com as saídas digitais ligadas a bornes-relê para isolação. Resistores instalados nos módulos A39.1 . . . A39.4

[2]Projeto número 035815E/05 de outubro de 2005, fonte: Weg Automação S.A.

Tabela 5.12: Influência do seccionamento de conduítes para o modelo PA2

Seccionamento	Custo total (R$)	Conduítes	Nós	Tempo (s)
500	110,22	30	24	0,156
250	97,98	38	32	0,233
100	94,26	61	55	0,592
75	93,49	76	70	0,945
50	93,21	99	93	1,625
30	93,14	153	147	3,878
20	93,22	221	215	8,516
10	93,13	427	421	32,562
5	93,07	831	825	136,825
1	93,04	4063	4057	3273,365

são usados para converter o sinal de corrente das saídas analógicas (0-20 mA) para o sinal de tensão exigido pelos equipamentos de campo. O painel é alimentado por um transformador de comando (T1) e fontes de alimentação reguladas de 24 Vcc (A3.1 . . . A3.4). Os componentes do painel são instalados em duas placas de montagem em configuração *back to back*, dispostos conforme a Figura 5.10.

Todas as interligações conectam exatamente dois terminais, logo, este painel não permite comparar o desempenho relativo da busca exaustiva, algoritmo guloso e otimização por colônia de formigas. Este modelo também foi objeto de estudo das versões simplificadas do algoritmo de roteamento, que não previam estes algoritmos de roteamento – Algoritmo 3.1 da Seção 3.5.2 e Algoritmo 3.2 da Seção 3.5.2.

Os dados dos conduítes e da disposição física dos terminais foram extraídos de um modelo mecânico tridimensional do painel elaborado em AutoCAD (Figura 5.11) por um aplicativo *ad-hoc* desenvolvido na linguagem de programação AutoLISP. Os dados da fiação foram extraídos da lista de cablagem originalmente usada para a montagem do painel, no total de 692 cabos conectando 1096 terminais únicos. A Tabela 5.13 lista os tipos de cabos usados neste painel.

A passagem dos cabos entre as faces frontal e posterior do painel é feita através de dez

Tabela 5.13: Dados dos cabos usados para o Painel PA3

Cor / Tipo	Bitola (mm^2)	Seção transversal externa (mm^2)	Custo (R$/m)
2 vias, blindado	0,5	28,27	1,69
Azul escuro	0,75	3,80	0,26
Preto	0,75	3,80	0,26
Vermelho	0,75	3,80	0,26
Amarelo	1,5	6,16	0,37
Preto	1,5	6,16	0,37
Verde/Amarelo	1,5	6,16	0,36
Preto	2,5	9,08	0,58
Verde/Amarelo	2,5	9,08	0,58
Verde/Amarelo	4	11,95	0,90

rasgos oblongos nas placas de montagem e nas paredes posteriores das canaletas, modelados como conduítes do tipo fechado. Alguns segmentos de canaleta foram reservados para alojar os cabos conectando o painel aos equipamentos em campo, instalados pelo cliente, e não foram modelados.

5.2.3.1 Teste das Distâncias de Seccionamento de Conduítes

A Tabela 5.14 apresenta os resultados obtidos com diversos valores de seccionamento de conduítes. O teste foi executado usando o algoritmo de inserção heurística com *first fail* e, como todas as interligações deste painel conectam exatamente dois terminais, nenhum algoritmo de roteamento de interligações foi executado. O procedimento foi executado para cada valor de seccionamento indicado na coluna “Seccionamento”, resultando em um modelo derivado com o número de conduítes e nós indicados nas respectivas colunas. O tempo de execução indicado é a média aritmética dos valores medidos em cinco execuções do aplicativo para cada valor de seccionamento. Este procedimento destina-se exclusivamente a reduzir eventuais imprecisões na tomada dos tempos, pois, como os algoritmos usados neste teste são determinísticos, não há

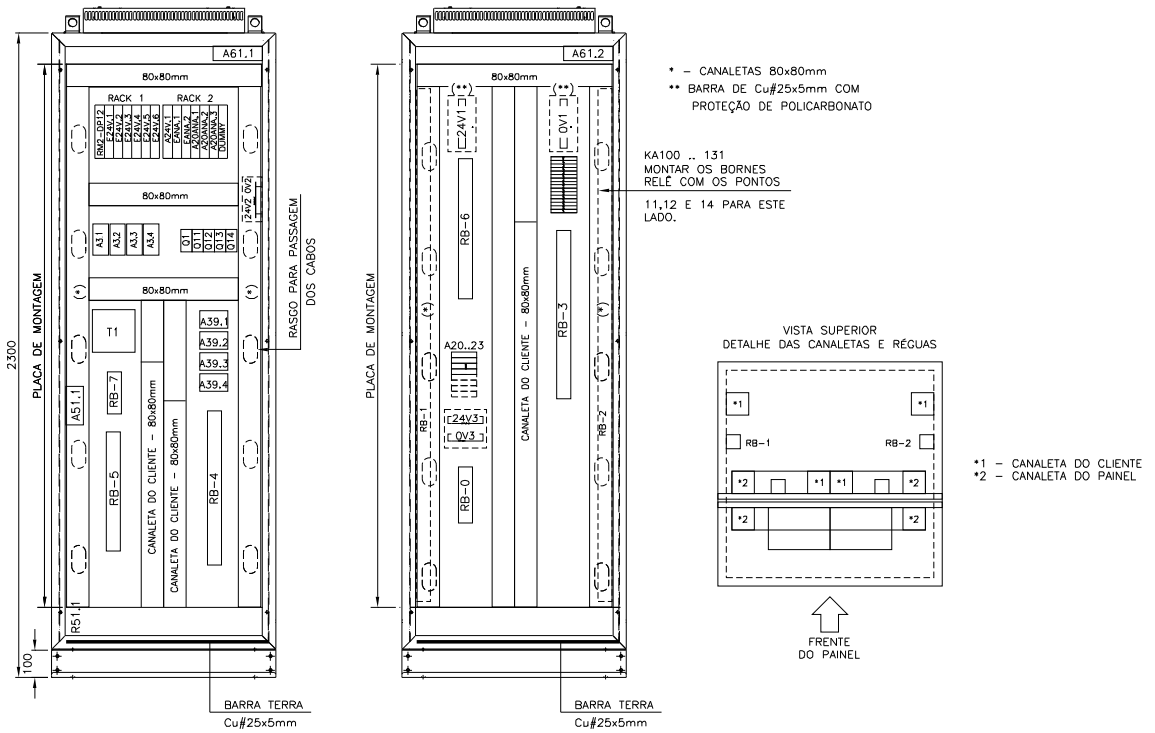


Figura 5.10: *Layout* mecânico do Painel PA3 (Fonte: WEG Automação S.A.)

diferenças nos resultados entre as execuções. Todas as execuções para o seccionamento de 1 mm foram interrompidas quando o tempo de execução ultrapassou os 180 minutos.

5.2.3.2 Comparação com Resultados Fornecidos por um Especialista Humano

O modelo do Painel PA3 deriva de um painel construído fisicamente, cuja cablagem foi especificada por um especialista humano na época da sua construção. Este fato permite a comparação direta entre os resultados fornecidos pelos algoritmos de roteamento e os dados existentes sobre a construção do painel. A Tabela 5.15 apresenta a quantidade de cabos especificados pelo especialista e a quantidade gerada pela execução do algoritmo de roteamento por inserção heurística com *first fail* (FF) e distância de seccionamento de conduítes igual a 50 mm. Todas as interligações possuem exatamente dois terminais, logo, há apenas uma possibilidade para a sequência de conexão e não há influência do algoritmo da etapa de roteamento (busca exaustiva). Os dados do custo dos cabos derivam dos valores listados na Tabela 5.13.

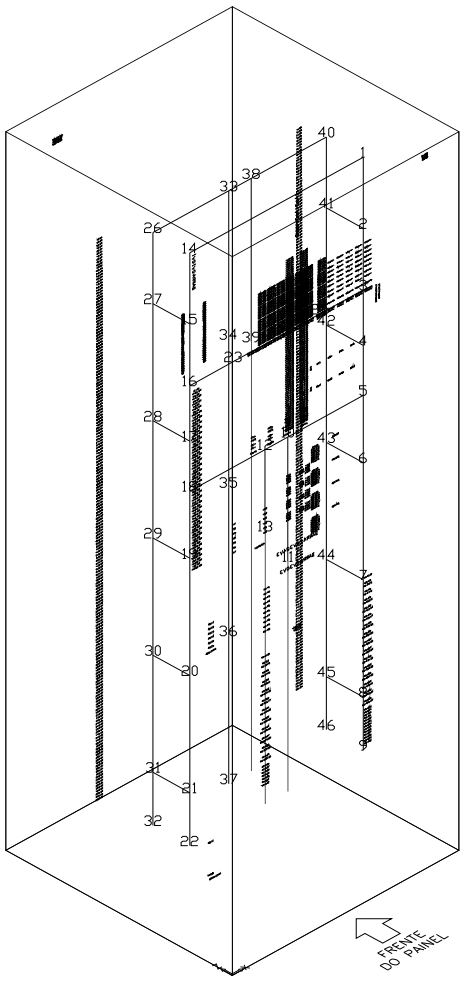


Figura 5.11: Modelo CAD usado para extração dos dados do Painel PA3

5.2.3.3 Comparação entre os Interpretadores Lua, LuaJIT e LuaJIT2

O teste comparativo entre os interpretadores Lua, LuaJIT e LuaJIT2 objetiva determinar a influência do interpretador Lua sobre a implementação dos algoritmos. O objetivo deste teste não é determinar as características dos algoritmos implementados, mas fornecer dados que permitam estimar a influência do interpretador sobre os demais testes.

O procedimento de teste consiste em repetir os experimentos executados originalmente com o interpretador Lua 5.1.4 (descritos na Seção 5.2.3.1) com os interpretadores *just-in-time* LuaJIT 1.1.5 e LuaJIT 2.0 beta 2[3]. A Tabela 5.16 lista os resultados obtidos. A coluna "Lua 5.1.4" apresenta uma cópia dos tempos de execução do teste da Seção 5.2.3.1, contrastados com os

[3]No momento conclusão deste trabalho, o LuaJIT2 ainda está em desenvolvimento e algumas funcionalidades previstas ainda não estão disponíveis. Entretanto, os recursos já disponíveis e os bons resultados divulgados em `http://luajit.org/status.html` motivaram sua a inclusão nos testes

Tabela 5.14: Influência do seccionamento de conduítes para o modelo PA3

Seccionamento	Custo total (R$)	Conduítes	Nós	Tempo (s)
500	338,15	60	48	0,885
250	354,24	84	72	1,846
100	354,96	160	148	7,423
75	351,79	220	208	14,483
50	352,55	300	288	28,028
30	351,99	470	458	75,455
20	351,96	687	675	179,645
10	351,91	1349	1337	672,567
5	351,97	2666	2654	2679,971
1	–	13164	13152	–

tempos obtidos com os dois interpretadores *just-in-time* em suas respectivas colunas. A coluna "V.R." representa a velocidade relativa do interpretador em relação ao interpretador Lua 5.1.4. O tempo de execução indicado é a média aritmética dos valores medidos em cinco execuções de cada teste. Este procedimento destina-se exclusivamente a reduzir eventuais imprecisões na tomada dos tempos, pois, como os algoritmos usados neste teste são determinísticos, não há diferenças nos resultados entre as execuções. Todas as execuções do interpretador LuaJIT 1.1.5 para o seccionamento de 1 mm foram interrompidas quando o tempo de execução ultrapassou os 60 minutos.

5.3 Avaliação dos Resultados

Os resultados obtidos nos testes com o protótipo desenvolvido para o tratamento do Problema do Roteamento de Cabos em Painéis Elétricos confirmam os resultados esperados em função dos critérios adotados durante o desenvolvimento. Os melhores resultados foram obtidos com as funções heurísticas desenvolvidas especialmente para esta aplicação e, em especial, com as três instâncias práticas descritas nas Seções 5.2.1, 5.2.2 e 5.2.3, que demonstram a viabilidade dos algoritmos desenvolvidos para aplicações industriais.

Tabela 5.15: Comparação entre os resultados fornecidos pelo algoritmo de roteamento e os resultados obtidos por um especialista humano para o Painel PA3

Cabo	Especialista		Algoritmo (FF+Ex)	
	(m)	R$	(m)	R$
4,0 mm^2 verde/amarelo	20,00	18,00	1,86	1,67
2,5 mm^2 preto	30,00	17,40	12,86	7,46
2,5 mm^2 verde/amarelo	30,00	17,40	15,97	9,26
1,5 mm^2 amarelo	30,00	11,10	11,37	4,21
1,5 mm^2 preto	100,00	37,00	85,56	31,66
1,5 mm^2 verde/amarelo	40,00	14,40	43,66	15,72
0,75 mm^2 preto	100,00	26,00	23,99	6,24
0,5 mm^2 duas vias blindado	100,00	169,00	77,22	130,50
0,75 mm^2 azul escuro	500,00	130,00	511,22	132,92
0,75 mm^2 vermelho	90,00	23,40	49,71	12,92
Custo total:		463,70		352,55

5.3.1 Avaliação dos Algoritmos de Inserção de Interligações

Os testes realizados indicam a superioridade da Inserção com *First Fail* face à Inserção Heurística Simples, um fato esperado em função dos critérios adotados no desenvolvimento deste algoritmo. Um fato não-antecipado foi a ineficiência da Inserção por Algoritmos Genéticos no teste descrito na Seção 5.1.3, especialmente quando comparada à Inserção com *First Fail*. Este fato é justificado face à consideração de que a IFF foi desenvolvida especialmente para lidar com as características comumente encontradas em painéis, enquanto a Inserção por Algoritmos Genéticos usa uma abordagem genérica para problemas de sequenciamento.

Em função destes resultados, adota-se a função de Inserção com *First Fail* como algoritmo padrão para a etapa de inserção de interligações, adotando-se a Inserção por Algoritmos Genéticos apenas para as instâncias que exijam a otimização de conduítes fortemente saturados.

Tabela 5.16: Influência do interpretador Lua nos tempos de execução para o modelo PA3

Secc.	Conduítes	Nós	Lua 5.1.4	LuaJIT 1.1.5		LuaJIT 2.0 β 2	
			Tempo (s)	Tempo (s)	V.R.	Tempo (s)	V.R.
500	60	48	0,885	0,571	1,549	0,103	5,535
250	84	72	1,846	1,172	1,575	0,143	8,184
100	160	148	7,423	4,714	1,575	0,412	11,443
75	220	208	14,483	9,263	1,564	0,710	13,039
50	300	288	28,028	18,191	1,541	1,266	14,373
30	470	458	75,455	47,954	1,573	2,846	16,847
20	687	675	179,645	111,774	1,607	6,345	17,617
10	1349	1337	672,567	439,519	1,530	24,716	17,783
5	2666	2654	2679,971	1729,173	1,550	105,545	16,383
1	13164	13152	–	–	–	2928,847	–

5.3.2 Avaliação dos Algoritmos de Roteamento

Os testes executados indicam que o algoritmo de roteamento por Busca Exaustiva (Ex) se mantém competitivo com os demais algoritmos avaliados em instâncias reduzidas da etapa de roteamento (até seis terminais). Este algoritmo torna-se ineficiente em instâncias com sete ou mais terminais e inviável em instâncias com nove ou mais terminais. O algoritmo de Busca Gulosa (BG) mantém, como esperado, o tempo de execução linear em função do número de terminais. Embora esta técnica seja incapaz de garantir a solução ótima para uma dada interligação, testes executados com instâncias práticas do problema demonstram que boas soluções são geradas – este fato deve-se principalmente às características de projeto comumente encontradas em painéis. O roteamento com Otimização por Colônias de Formigas mantem as características esperadas, embora o tempo de preparação do algoritmo torne-o pouco competitivo para instâncias com poucos terminais.

Estes resultados indicam que a melhor abordagem padrão para a etapa de roteamento é uma estratégia híbrida que aplique a Busca Exaustiva para instâncias com até seis terminais e a Busca Gulosa para as demais. O roteamento por Colônia de Formigas pode ser usado para

instâncias específicas por indicação do usuário. Esta estratégia híbrida também foi implementada no protótipo.

5.3.3 Seccionamento de Conduítes

Uma análise dos valores apresentados nas Tabelas 5.10, 5.12 e 5.14 permite deduzir a influência da distância de seccionamento dos conduítes sobre o tempo de execução dos algoritmos e o custo total da fiação que, por sua vez, pode ser extrapolado para a precisão na determinação do comprimento dos cabos do painel. As instâncias práticas do PRCPE apresentam um limite empírico na aplicabilidade das distâncias de seccionamento: a medida que este valor se aproxima da distância média entre os terminais, menor é sua influência sobre o comprimento dos cabos e, consequentemente, sobre o custo da fiação do painel. Esta relação entre a distância de seccionamento, distorção da solução e custo da fiação dos painéis PA1, PA2 e PA3 pode ser vista no gráfico da Figura 5.12, onde o eixo horizontal representa a distância de seccionamento dos conduítes e o eixo vertical representa a relação entre o custo da fiação gerada com este valor de seccionamento e o custo da fiação gerada com o menor valor de seccionamento disponível.

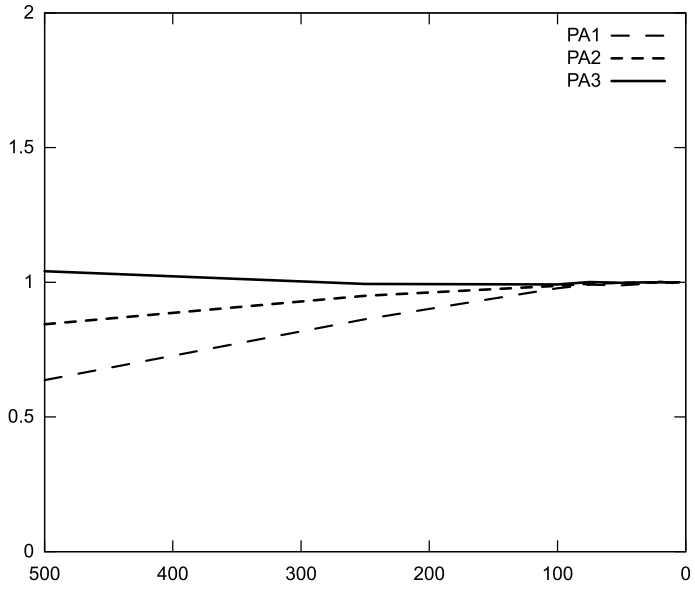


Figura 5.12: Relação entre a distância de seccionamento e custo relativo da fiação dos painéis PA1, PA2 e PA3

A Figura 3.10 torna este comportamento da distância de seccionamento dos conduítes mais explícito: pode-se notar a grande distorção na qualidade da solução para valores elevados (acima

de 100 mm), uma redução considerável desta distorção nos valores intermediários (100 a 50 mm) e pouca ou nenhuma variação para os valores inferiores a 50 mm.

Esta propriedade acaba por ressaltar o papel da distância de seccionamento dos conduítes como um parâmetro de seleção da precisão desejada para o cálculo da quantidade de cabos, ou seja, como um recurso adicional que permite ao usuário escolher um compromisso ideal entre o tempo de execução e a qualidade da solução.

5.3.4 Tempos de Execução

Os tempos de execução medidos para as diversas instâncias do Problema do Roteamento de Cabos em Painéis Elétricos analisadas nos testes refletem o comportamento esperado de crescimento íngreme em função da complexidade do modelo, especialmente quando opta-se por intervalos de seccionamento reduzidos. Estes tempos de execução elevados, porém, não limitam a aplicação da solução proposta às instâncias **práticas** do PRCPE para as quais os algoritmos foram projetados.

5.3.5 Influência do Interpretador Lua e da Linguagem de Programação

Os testes executados com os diferentes interpretadores Lua listados na Tabela 5.16 permitem separar a influência da linguagem de programação e do ambiente de execução das características fundamentais do algoritmo. Em uma consideração à parte, nota-se a diferença entre o crescimento do tempo de execução em função da complexidade do modelo para o interpretador LuaJIT2 que, nas instâncias mais complexas, chega a ser 16 vezes mais eficiente que a implementação de referência. Como nenhuma versão do protótipo proposto foi implementada em uma linguagem de programação com execução nativa (sem uma máquina virtual ou *bytecode*) para comparação, não é possível afirmar que haverá um ganho excepcional de velocidade com esta abordagem, porém, pode-se afirmar que o LuaJIT2 fornece excelentes tempos de execução para os cenários de uso previstos para este algoritmo.

5.4 Resumo do Capítulo

O protótipo desenvolvido para avaliação dos algoritmos propostos no Capítulo 4 foi submetido a dois conjuntos de testes: (1) testes artificiais, desenvolvidos para explorar características conhecidas dos algoritmos, ainda que incomuns em aplicações práticas e (2) testes práticos, que

simulam as condições reais encontradas em projetos industriais. Neste segundo grupo, destaca-se uma instância extraída de um projeto real que foi montado fisicamente.

Para a Etapa de Inserção, os melhores resultados foram obtidos com o algoritmo de Inserção com *First Fail*, algo coerente com as expectativas iniciais, visto que este algoritmo foi desenvolvido especialmente para explorar as características comumente encontradas em painéis industriais. Para a Etapa de Roteamento, adotou-se uma estratégia híbrida usando os algoritmos de Busca Exaustiva (para pequenas instâncias) e Busca Gulosa (para todas as demais instâncias) de modo a aproveitar as características vantajosas das duas abordagens.

Os algoritmos evolutivos de Inserção por Algoritmos Genéticos e Roteamento por Colônias de Formigas são aplicados aos casos restritos que exigem otimizações especiais e como alternativas para as instâncias onde os algoritmos heurísticos são incapazes de obter bons resultados.

6 Conclusão e Trabalhos Futuros

O trabalho aqui apresentado envolveu o estudo do processo de roteamento de cabos em painéis elétricos, sua formalização no Problema do Roteamento de Cabos em Painéis Elétricos descrito no Capítulo 3 e a aplicação de técnicas tradicionais da Inteligência Artificial e da Computação Evolucionária para a elaboração de algoritmos eficientes para tratamento deste problema e sua implementação em *software*.

A formalização do Problema do Roteamento de Cabos em Painéis Elétricos permitiu o estudo das suas principais características, a identificação das estratégias ótimas para abordagem dos seus subproblemas e a seleção de um subconjunto de características passíveis de exploração por meios heurísticos. Destes estudos, derivam os processos de tratamento de conduítes do tipo "aberto" e de saturação parcial de conduítes e da abordagem computacional para o PRCPE descrita na Seção 3.3.

Seis algoritmos foram desenvolvidos para o tratamento de subproblemas do PRCPE, incluindo a aplicação da Otimização por Colônias de Formigas com $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* e de um Algoritmo Genético. A aplicação do conhecimento gerado a partir da análise do problema e dos algoritmos idealizados culminou na elaboração de um aplicativo eficaz e eficiente para a resolução de instâncias reais do PRCPE, derivadas de projetos industriais. A viabilidade deste aplicativo foi comprovada por meio de testes artificiais (para validação de características conhecidas) e reais (para análise do comportamento prático do *software*), descritas no Capítulo 5.

Através dos testes realizados conclui-se que:

- A distância de seccionamento de conduítes é um parâmetro para escolha da qualidade da solução;
- Há uma distância de seccionamento ideal, abaixo da qual a maior complexidade do modelo derivado não implica em uma solução melhor;
- Os algoritmos de inserção de interligações que exploram as características específicas encontradas nos grafos de painéis, propostos e implementados neste trabalho, apresentam desempenho superior aos algoritmos generalistas (AG e ACO);

- É viável a aplicação de um algoritmo de busca exaustiva na etapa de roteamento para a obtenção da solução ótima em instâncias com até seis terminais;

- O roteamento por Busca Gulosa gerou boas soluções com execução em tempo aceitável para a aplicação proposta em instâncias com sete ou mais terminais;

6.1 Trabalhos Futuros

Os tópicos sujeitos a abordagem para continuidade dos estudos sobre o Problema do Roteamento de Cabos em Painéis Elétricos dividem-se em duas categorias: extensões do modelo do problema em consideração a novos fatores técnicos e novas abordagens para a sua implementação e solução. Para o primeiro grupo, cita-se:

- A inclusão de considerações sobre restrições mecânicas no percurso dos cabos entre os pontos de entrada e o terminais dos componentes e sua influência no comprimento destes segmentos de cabo;

- A consideração do raio de curvatura mínimo no cálculo do comprimento dos cabos e sua influência na ocupação dos conduítes;

- A inclusão de restrições ao compartilhamento de conduítes por determinados tipos de cabo que permitam, por exemplo, a designação de conduítes específicos para cabos do circuito de comando, cabos de força, cabos de sinal sujeitos a interferências e PELV (*protected extra-low voltage*);

- A exigência de distâncias mínimas entre determinados cabos do modelo;

- A representação de critérios de compatibilidade eletromagnética entre cabos e componentes no modelo do painel e consideração destes critérios na solução do problema. Esta mudança está relacionada aos critérios supra-citados de segregação de cabos e distâncias mínimas, mas também implica na adição de novas propriedades para conduítes blindados e a representação de blindagens em painéis;

- A adição de exceções e restrições excepcionais para a geração dos cabos de uma interligação e sua representação no modelo do painel. Este recurso permitiria, por exemplo, a especificação de terminais que aceitam apenas um ou mais de dois cabos. Esta mudança

também altera as características do problema da determinação da ordem de conexão dos terminais na Etapa de Roteamento (Seção 4.2), que deixa de ser um PCV clássico.

Como sugestões para novas pesquisas na implementação de soluções para o modelo proposto, cita-se:

- Estudar a viabilidade de derivar o modelo do painel da sua representação no formato definido na norma ISO 10303:212;
- O estudo de métodos para determinar automaticamente a distância de seccionamento ideal para uma instância do problema;
- A avaliação das outras estratégias para seccionamento de conduítes descritas na Seção 3.2 e o estudo de novas abordagens para este problema;
- O estudo de alternativas para a recuperação de um modelo inviável, isto é, de métodos que, na presença de uma instância do problema identificada como sem solução, proponham o menor conjunto de alterações necessárias para torná-la viável;
- O estudo de novas abordagens para etapa de inserção de interligações e, em especial, de modelos mais eficientes para a Inserção por Algoritmos Genéticos;
- O estudo de alternativas mais eficientes para a determinação do ponto de entrada mais próximo de um dado terminal usando, por exemplo, técnicas para busca de vizinhanças e *hashing* sensível a localidade;
- O estudo de técnicas que permitam detectar a independência mútua entre duas ou mais interligações de modo a permitir a paralelização da Etapa de Roteamento e melhor aproveitamento de processadores com vários núcleos;

Referências

APPLEGATE, D. L.; BIXBY, R. E.; CHVATAL, V.; COOK, W. J. *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton University Press, 2007. 606 p. ISBN 0691129932, 9780691129938.

BENT, R. W.; HENTENRYCK, P. van. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 52, n. 6, p. 977–987, 2004. ISSN 0030-364X.

BOSCH. *CL200 Manual Ed. 102 – 1070-072-145-102*. Erbach, Alemanha: Robert Bosch GmbH, 2001.

BRYANT, R. E.; O'HALLARON, D. R. *Computer Systems: A Programmer's Perspective*. Upper Saddle River, NJ, USA: Prentice Hall, 2002. ISBN 978-0130340740.

BULLNHEIMER, B.; HARTL, R. F.; STRAUSS, C. A new rank based version of the ant system - a computational study. *Central European Journal for Operations Research and Economics*, v. 7, p. 25–38, 1997.

CASEAU, Y.; LABURTHE, F. Solving small TSPs with constraints. In: NAISH, L. (Ed.). *Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming*. Leuven, Belgium: MIT Press, 1997. (MIT Press Series in Logic Programming), p. 316–330. ISBN 0-262-64035-X. ISSN 1061-0464.

CASTILLO, O.; MELIN, P.; MONTIEL, O.; SEPÚLVEDA, R. Application of a breeder genetic algorithm for system identification in an adaptive finite impulse response filter. *Proceedings of the the 3rd NASA/DoD Workshop on Evolvable Hardware*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 146–153, 2001.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. Distributed optimization by ant colonies. In: VARELA, F. J.; BOURGINE, P. (Ed.). *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. Cambridge, MA, USA: MIT Press, 1992. p. 134–142.

CONRU, A. A genetic approach to the cable harness routing problem. *Proceedings of the First IEEE Conference on Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence*, Orlando, FL, USA, v. 1, n. 1, p. 200–205, June 1994.

CORDEAU, J.-F.; GENDREAU, M.; LAPORTE, G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, Montréal, Canada, v. 30, n. 2, p. 105–119, 1997. ISSN 1097-0037.

CORDEAU, J.-F.; LAPORTE, G. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, v. 1, n. 2, p. 89–101, 2003.

CORDÓN, O.; HERRERA, F.; STÜTZLE, T. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, v. 9, p. 141–175, 2002.

CORDÓN, O.; VIANA, I. naki Fernández de; HERRERA, F. Analysis of the best-worst ant system and its variants on the tsp. *Mathware & Soft Computing*, v. 9, n. 3, p. 177–192, 2002. ISSN 1134-5632 / 1989-533X.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms*. 2. ed. Cambridge, Massachusetts, USA: The MIT Press, 2001. 1184 p. ISBN 0-262-03293-7.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management Science*, INFORMS, v. 6, n. 1, p. 80–91, 1959. ISSN 00251909.

DENEUBOURG, J.; PASTEELS, J.; VERHAEGHE, J. Probabilistic behaviour in ants: A strategy of errors? *Journal of Theoretical Biology*, v. 105, n. 2, p. 259–271, 1983. ISSN 0022-5193.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, v. 1, p. 269–271, 1959.

DOOMS, G.; DEVILLE, Y.; DUPONT, P. Constrained metabolic network analysis: discovering pathways using CP(Graph). In: *Proceedings of the Workshop on Constraint Based Methods for Bioinformatics*. Sitges, Barcelona, Spain: [s.n.], 2005.

DORIGO, M.; CARO, G. D.; GAMBARDELLA, L. M. Ant algorithms for discrete optimization. *Artificial Life*, v. 5, n. 2, p. 137–172, 1999.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, p. 53–66, 1997.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v. 26, p. 29–41, 1996.

EPLAN. *EPLAN Cabinet*. 2009. Disponível em: <http://www.eplan.de/products/electrical-engineering/eplan-cabinet.html>. Acesso em: Novembro de 2009.

FIGUEIREDO, L. H. de. *lrandom*. 2009. Disponível em: <http://www.tecgraf.puc-rio.br/~lhf/ftp/lua/>. Acesso em: Novembro de 2009.

FOGEL, D. B. Nils barricelli – artificial life, coevolution, self-adaptation. *Computational Intelligence Magazine, IEEE*, v. 1, n. 1, p. 41–45, Feb 2006. ISSN 1556-603X.

GAMBARDELLA, L. M.; DORIGO, M. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In: PRIEDITIS, A.; RUSSELL, S. (Ed.). *Machine Learning: Proceedings of the Twelfth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1995. p. 252–260.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1. ed. Boston, MA, USA: Addison-Wesley Professional, 1989. 372 p. ISBN 0201157675.

GOSS, S.; ARON, S.; DENEUBOURG, J.; PASTEELS, J. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, Springer-Verlag, v. 76, n. 12, p. 579–581, December 1989.

GREFENSTETTE, J. J.; GOPAL, R.; ROSMAITA, B. J.; GUCHT, D. van. Genetic algorithms for the traveling salesman problem. In: GREFENSTETTE, J. J. (Ed.). *Proceedings of the 1st International Conference on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1985. p. 160–168. ISBN 0-8058-0426-9.

IERUSALIMSCHY, R.; FIGUEIREDO, L. H. de; CELES, W. Lua: an extensible extension language. *Software: Practice & Experience*, v. 26, n. 6, p. 635–652, 1996.

ITTNER, A. E.; SÁ, C. C. de; SASSE, F. D. A heuristic approach to the cable routing problem in electrical panels. In: *Proceedings of the VI Congress of Logic Applied to Technology – LAPTEC'07*. Santos, São Paulo, Brasil: [s.n.], 2007. ISBN 978-85-99561-45-4.

ITTNER, A. E.; SÁ, C. C. de; SASSE, F. D. An improved heuristic solution to the cable routing problem in electrical panels. In: *II Workshop of Computational Intelligence - WCI'2008*. Salvador, Bahia, Brasil: [s.n.], 2008. p. 75–80.

ITTNER, A. E.; SÁ, C. C. de; SASSE, F. D. A heuristic approach to the cable routing problem in electrical panels. In: LAMBERT-TORRES, G.; ABE, J. M.; FILHO, J. I. da S.; MARTINS, H. G. (Ed.). *Advances in Techological Applications of Logical and Intelligent Systems – Selected Papers from the Sixth Congress on Logic Applied to Technology*. Amsterdam, Netherlands: IOS Press, 2009. (Frontiers in Artificial Intelligence and Applications, v. 186), p. 55–66. ISBN 978-1-58603-936-3. ISSN 0922-6389.

JONG, K. A. de; SPEARS, W. M. Using genetic algorithm to solve $\mathcal{NP}$-complete problems. In: SCHAFFER, J. D. (Ed.). *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA, USA: Morgan Kaufmann, 1989. p. 124–132.

KABUL, I.; GAYLE, R.; LIN, M. C. Cable route planning in complex environments using constrained sampling. In: . New York, NY, USA: ACM Press, 2007. p. 395–402. ISBN 978-1-59593-666-0.

KLOSKE, D. A.; SMITH, R. E. *Bulk Cable Routing Using Genetic Algorithms*. Tuscaloosa, AL, USA, 1994.

KRAJCAR, S.; SKRLEC, D.; PRIBICEVIC, B.; BLAGAJAC, S. GA approach to solving multiple vehicle routing problem. In: PINTO-FERREIRA, C.; MAMEDE, N. J. (Ed.). *Progress in Artificial Intelligence: Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA'95)*. London, UK: Springer-Verlag, 1995. (Lecture Notes in Computer Science, v. 990), p. 473–481. ISBN 3-540-60428-6.

LARRAÑAGA, P.; KUIJPERS, C.; MURGA, R.; INZA, I.; DIZDAREVIC, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, Springer Netherlands, v. 13, n. 2, p. 129–170, April 1999. ISSN 0269-2821/1573-7462.

LETCHFORD, A. N.; LYSGAARD, J.; EGLESE, R. W. A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, v. 58, n. 12, p. 1642–1651, December 2007.

LOPES, H. S. Algoritmos genéticos em projetos de engenharia: aplicações e perspectivas futuras. In: *Anais do IV Simpósio Brasileiro de Automação Inteligente*. São Paulo, SP, Brasil: Universidade de São Paulo, 1999. p. 64–74.

MA, X.; IIDA, K.; XIE, M.; NISHINO, J.; ODAKA, T.; OGURA, H. A genetic algorithm for the optimization of cable routing. *Systems and Computers in Japan*, Wiley-Interscience, New York, NY, USA, v. 37, n. 7, p. 61–71, 2006.

MANIEZZO, D.; DORIGO, M.; MANIEZZO, V.; COLORNI, A. *The Ant System: An Autocatalytic Optimizing Process*. Brussels, Belgium, 1991.

MANIEZZO, V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 11, n. 4, p. 358–369, 1999. ISSN 1526-5528.

MARTELLO, S.; TOTH, P. *Knapsack Problems: Algorithms and Computer Implementations*. West Sussex, England: John Wiley & Sons, 1990. 308 p. (Wiley-Interscience series in discrete mathematics and optimization). ISBN 0-471-92420-2.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, ACM, New York, NY, USA, v. 8, n. 1, p. 3–30, 1998. ISSN 1049-3301.

MAUCHER, M. *On the influence of non-perfect randomness on probabilistic algorithms*. Tese (Doutorado) — Universität Ulm, Institut für Theoretische Informatik, Ulm, Alemanha, 2009.

MÜHLENBEIN, H.; SCHLIERKAMP-VOOSEN, D. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary Computation*, MIT Press, Cambridge, MA, USA, v. 1, n. 1, p. 25–49, 1993. ISSN 1063-6560.

OLIVEIRA, H. C. B. de; VASCONCELOS, G. C. A hybrid search method for the vehicle routing problem with time windows. In: *Annals of Operations Research*. Netherlands: Springer, 2008. v. 180, n. 1, p. 125–144. ISSN 0254-5330/1572-9338.

PALACIOS, H.; GEFFNER, H. Constrained metabolic network analysis: discovering pathways using CP(Graph). In: *XVIII Latin-American Conference on Informatics (CLEI-2002)*. Montevideo, Uruguay: [s.n.], 2002.

PALL, M. *The LuaJIT Project*. 2009. Disponível em: <http://www.luajit.org/>. Acesso em: Novembro de 2009.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. Mineola, NY, USA: Dover Publications, 1998. 512 p. ISBN 0-486-40258-4.

PELLEGRINI, P.; FAVARETTO, D.; MORETTI, E. On $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*'s parameters. In: DORIGO, M.; BIRATTARI, M.; STÜTZLE, T.; GAMBARDELLA, L. M.; MARTINOLI, A.; POLI, R. (Ed.). *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence – ANTS'2006*. Berlin, Deutschland: Springer, 2006. (Lecture Notes in Computer Science, v. 4150), p. 203–214. ISBN 3-540-38482-0. ISSN 0302-9743/1611-3349.

PESANT, G.; GENDREAU, M.; POTVIN, J.-Y.; ROUSSEAU, J.-M. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 32, n. 1, p. 12–29, 1998. ISSN 1526-5447.

PESANT, G.; GENDREAU, M.; ROUSSEAU, J.-M. GENIUS-CP: a generic single-vehicle routing algorithm. In: SMOLKA, G. (Ed.). *Proceedings of the Third International Conference in Principles and Practice of Constraint Programming (CP'97)*. Berlin, Germany: Springer, 1997. (Lecture Notes in Computer Science, v. 1330), p. 420–434. ISBN 3-540-63753-2.

PISINGER, D. Where are the hard knapsack problems? *Computers & Operations Research*, Elsevier Science Ltd., Oxford, UK, v. 32, n. 9, p. 2271–2284, 2005. ISSN 0305-0548.

PISINGER, D.; RØPKE, S. A general heuristic for vehicle routing problems. *Computers & Operations Research*, v. 34, n. 8, p. 2403–2435, 2007.

REINELT, G. *TSPLIB*. 2009. Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>. Acesso em: Novembro de 2009.

RIZZOLI, A. E.; MONTEMANNI, R.; LUCIBELLO, E.; GAMBARDELLA, L. M. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, v. 1, n. 2, p. 135–151, 2007.

SHAW, P. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science; Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, Springer-Verlag, v. 1520, p. 417–431, 1998.

STÜTZLE, T.; HOOS, H. H. $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System* and local search for the traveling salesman problem. In: *IEEE International Conference on Evolutionary Computation, 1997*. Indianapolis, IN, USA: IEEE Publishing, 1997. p. 309–314. ISBN 0-7803-3949-5.

STÜTZLE, T.; HOOS, H. H. $\mathcal{MAX}$–$\mathcal{MIN}$ *Ant System*. *Future Generation Computer Systems*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, v. 16, n. 9, p. 889–914, 2000. ISSN 0167-739X.

VASKO, F. J.; BARBIERI, R. S.; RIEKSTS, B. Q.; REITMEYER, K. L.; STOTT, J. K. L. The cable trench problem: combining the shortest path and minimum spanning tree problems. *Computers & Operations Research*, Elsevier Science, Oxford, UK, v. 29, n. 5, p. 441–458, 2002. ISSN 0305-0548.

VIEGAS, R.; AZEVEDO, F. GRASPER: A Framework for Graph Constraint Satisfaction Problems. Guimarães, Portugal, December 2007.

ZUNKEN. *E^3.series*. 2009. Disponível em: <http://www.zuken.com/products/e3-series/system/electrical.aspx>. Acesso em: Novembro de 2009.

Anexo I – Modelo do Painel PA1

```
-- Cria iteradores para terminais.
local function Counter(prefix, sufix, from)
  return function()
    local ret = prefix .. from .. sufix
    from = from + 1
    return ret
  end
end


-- Régua de bornes horizontal. Dimensões: 5*nterms+10 x 50 x 50
-- Terminais numerados de 1 a nterms (A e B)
-- Exemplo:
--  term_strip_1(tbl, "X1", 4, 0, 0, 0)
--    ["X1:1A"]  = {  5, 45, 20 },
--    ["X1:1B"]  = {  5,  5, 20 },
--    ["X1:2A"]  = { 10, 45, 20 },
--    ["X1:2B"]  = { 10,  5, 20 },
--    ["X1:3A"]  = { 15, 45, 20 },
--    ["X1:3B"]  = { 15,  5, 20 },
--    ["X1:4A"]  = { 20, 45, 20 },
--    ["X1:4B"]  = { 20,  5, 20 },
--

local function term_strip_acme_1(tbl, tag, nterms, x, y, z)
  local terms = { }
  for n = 1, nterms do
    local t = tag .. ":" .. tostring(n)
    tbl[t .. "A"] = { 5*n + x, y + 45, z + 20 }
    tbl[t .. "B"] = { 5*n + x, y + 5,  z + 20 }
  end
end


-- Contator de força 2NA+2NF -- Medidas: 45x81x87 (LxAxP)
-- Modelado a partir de um CWM9.xx, CWM12.xx, CWM18.xx
local function cont_weg_cwm9_22(tbl, tag, x, y, z)
  tbl[tag .. ":13"] = { x + 6.8,            y + 63, z + 66 }
  tbl[tag .. ":14"] = { x + 6.8,            y + 16, z + 66 }
  tbl[tag .. ":21"] = { x + 6.8 + 10.425,   y + 63, z + 66 }
  tbl[tag .. ":22"] = { x + 6.8 + 10.425,   y + 16, z + 66 }
  tbl[tag .. ":33"] = { x + 6.8 + 2*10.425, y + 63, z + 66 }
  tbl[tag .. ":34"] = { x + 6.8 + 2*10.425, y + 16, z + 66 }
  tbl[tag .. ":43"] = { x + 6.8 + 3*10.425, y + 63, z + 66 }
  tbl[tag .. ":44"] = { x + 6.8 + 3*10.425, y + 16, z + 66 }

  tbl[tag .. ":1"]  = { x + 6.8,            y + 63, z + 39 }
```

```
  tbl[tag .. ":2"]  = { x + 6.8,            y + 16, z + 39 }
  tbl[tag .. ":3"]  = { x + 6.8 + 10.425,   y + 63, z + 39 }
  tbl[tag .. ":4"]  = { x + 6.8 + 10.425,   y + 16, z + 39 }
  tbl[tag .. ":5"]  = { x + 6.8 + 2*10.425, y + 63, z + 39 }
  tbl[tag .. ":6"]  = { x + 6.8 + 2*10.425, y + 16, z + 39 }
  tbl[tag .. ":A1"] = { x + 6.8 + 3*10.425, y + 63, z + 39 }
  tbl[tag .. ":A2"] = { x + 6.8 + 3*10.425, y + 16, z + 39 }
end


-- Disjuntor motor 1NA + 1NF -- Medidas: 45x97x77 (AxLxP)
-- Modelado a partir de um MPW16
local function cbr_weg_mpw1611(tbl, tag, x, y, z)
  -- Contatos principais
  tbl[tag .. ":1"] = { x + 8,           y + 82, z + 31 }
  tbl[tag .. ":2"] = { x + 8,           y + 6,  z + 31 }
  tbl[tag .. ":3"] = { x + 8 + 12.87,   y + 82, z + 31 }
  tbl[tag .. ":4"] = { x + 8 + 12.87,   y + 6,  z + 31 }
  tbl[tag .. ":5"] = { x + 8 + 2*12.87, y + 82, z + 31 }
  tbl[tag .. ":6"] = { x + 8 + 2*12.87, y + 6,  z + 31 }

  -- Bloco de contatos auxiliares ACBF11
  tbl[tag .. ":21"] = { x + 8.5, y + 72, z + 55 }
  tbl[tag .. ":22"] = { x + 17,  y + 72, z + 55 }
  tbl[tag .. ":13"] = { x + 27,  y + 72, z + 55 }
  tbl[tag .. ":14"] = { x + 36,  y + 72, z + 55 }
end


-- Disjuntor monopolar MDW -- Medidas: 17,9x79x65,4 (AxLxP)
local function cbr_weg_mdw_1p(tbl, tag, x, y, z)
  tbl[tag .. ":1"] = { x + 9, y + 73, z + 22 }
  tbl[tag .. ":2"] = { x + 9, y + 7,  z + 22 }
end


-- Disjuntor bipolar MDW -- Medidas: 38,5x79x65,4 (AxLxP)
local function cbr_weg_mdw_2p(tbl, tag, x, y, z)
  tbl[tag .. ":1"] = { x + 9,  y + 73, z + 22 }
  tbl[tag .. ":2"] = { x + 9,  y + 7,  z + 22 }
  tbl[tag .. ":3"] = { x + 26, y + 73, z + 22 }
  tbl[tag .. ":4"] = { x + 26, y + 7,  z + 22 }
end


-- Transformador de comando. Dimensões: 110x110x110 (AxLxP)
-- Conexões:   Primário = H1-H2    Secundário = X1-X2
local function trans_acme_1(tbl, tag, x, y, z)
  tbl[tag .. ":H1"] = { x + 50, y + 100, z + 105 }
  tbl[tag .. ":H2"] = { x + 60, y + 100, z + 105 }
  tbl[tag .. ":X1"] = { x + 50, y + 10,  z + 105 }
  tbl[tag .. ":X2"] = { x + 60, y + 10,  z + 105 }
end
```

```
-- CLP, dimensões: 190x100x70 (LxAxP)
--
--  Composição:
--
--  Parte superior:
--    16 entradas digitais isoladas com terminais numerados na
--      sequência I1A-I1B, I2A-I2B, ... I15A-I15B, I16A-I16B
--
--  Parte inferior:
--    16 saídas digitais a relê NA com terminais numerados na
--     sequência O1A-O1B, O2A-O2B, ... O15A-O15B, O16A-O16B
--
--   Alimentação: L1, L2
--

local function plc_acme_1(tbl, tag, x, y, z)

  -- Alimentação:
  tbl[tag .. ":L1"] = { x + 15, y + 95, z + 65 }
  tbl[tag .. ":L2"] = { x + 15, y +  5, z + 65 }

  -- Entradas e saídas:
  for i = 0, 15 do
    local t = tostring(i+1)
    tbl[tag .. ":I" .. t .. "A"] = { x + 10*i + 20, y + 95, z + 65 }
    tbl[tag .. ":I" .. t .. "B"] = { x + 10*i + 25, y + 95, z + 65 }
    tbl[tag .. ":O" .. t .. "A"] = { x + 10*i + 20, y + 5,  z + 65 }
    tbl[tag .. ":O" .. t .. "B"] = { x + 10*i + 25, y + 5,  z + 65 }
  end
end

local terms = {
  -- Barras de alimentação na saída do disjuntor principal
  ["A"] = { 408, 850, 300 },
  ["B"] = { 456, 850, 300 },
  ["C"] = { 500, 850, 300 },

  -- Barra terra do painel
  ["PE"] = { 40, 130, 50 }
}

term_strip_acme_1(terms, "X1", 50, 50, 335, 50)

cont_weg_cwm9_22(terms, "1K1",  60, 515, 50)
cont_weg_cwm9_22(terms, "2K1", 120, 515, 50)
cont_weg_cwm9_22(terms, "3K1", 180, 515, 50)
cont_weg_cwm9_22(terms, "4K1", 240, 515, 50)
cont_weg_cwm9_22(terms, "5K1", 300, 515, 50)

cbr_weg_mpw1611(terms, "1Q1",  60, 745, 50)
cbr_weg_mpw1611(terms, "2Q1", 120, 745, 50)
```

```
cbr_weg_mpw1611(terms, "3Q1", 180, 745, 50)
cbr_weg_mpw1611(terms, "4Q1", 240, 745, 50)
cbr_weg_mpw1611(terms, "5Q1", 300, 745, 50)

plc_acme_1(terms, "A1", 80, 975, 50)
cbr_weg_mdw_2p(terms, "Q2", 315, 985, 50)
cbr_weg_mdw_1p(terms, "Q3", 360, 985, 50)
trans_acme_1(terms, "T1", 400, 965, 50)


local conns = {
  -- Alimentação do comando
  { terms = { "A", "Q2:1" }, ctype = 2 },
  { terms = { "B", "Q2:3" }, ctype = 2 },

  { terms = { "Q2:2", "T1:H1" }, ctype = 2 },
  { terms = { "Q2:4", "T1:H2" }, ctype = 2 },

  { terms = { "T1:X1", "Q3:1" }, ctype = 2 }
}


-- Terminais ligados ao fase e ao neutro do comando. Esta tabela será
-- modificada pelo loop que criará os típicos e só será inserida na
-- lista de conexões após a conclusão desta etapa.

local terms_line = { "Q3:2", "A1:L1" }
local terms_neutral = { "T1:X2", "A1:L2" }

-- Sequência de terminais para os bornes da régua X1
local new_term_x1 = Counter("X1:", "A", 1)


for i = 1, 5 do

  -- Ligações de força saindo das barras de alimentação para o disjuntor
  conns[#conns+1] = { terms = { "A", i .. "Q1:1" }, ctype = 3 }
  conns[#conns+1] = { terms = { "B", i .. "Q1:3" }, ctype = 3 }
  conns[#conns+1] = { terms = { "C", i .. "Q1:5" }, ctype = 3 }

  -- Ligações de força saindo do disjuntor para o contator
  conns[#conns+1] = { terms = { i .. "Q1:2", i .. "K1:1" }, ctype = 2 }
  conns[#conns+1] = { terms = { i .. "Q1:4", i .. "K1:3" }, ctype = 2 }
  conns[#conns+1] = { terms = { i .. "Q1:6", i .. "K1:5" }, ctype = 2 }

  -- Ligações de força saindo do contator para a régua de bornes
  conns[#conns+1] = { terms = { i .. "K1:2", new_term_x1() }, ctype = 2 }
  conns[#conns+1] = { terms = { i .. "K1:4", new_term_x1() }, ctype = 2 }
  conns[#conns+1] = { terms = { i .. "K1:6", new_term_x1() }, ctype = 2 }
  conns[#conns+1] = { terms = { "PE", new_term_x1() }, ctype = 4 }

  -- Contato auxiliar do disjuntor numa entrada do CLP
  terms_line[#terms_line+1] = i .. "Q1:13"
```

```
  conns[#conns+1] = { terms = { i .. "Q1:14", "A1:I" .. i .. "A" },
                      ctype = 1 }
  terms_neutral[#terms_neutral+1] = "A1:I" .. i .. "B"

  -- Saída do CLP para a bobina do contator
  terms_line[#terms_line+1] = "A1:O" .. i .. "A"
  conns[#conns+1] = { terms = { "A1:O" .. i .. "B", i .. "K1:A1" },
                      ctype = 1 }
  terms_neutral[#terms_neutral+1] = i .. "K1:A2"

end

-- Adiciona neutro e fase à lista de conexões
conns[#conns+1] = { terms = terms_line, ctype = 1 }
conns[#conns+1] = { terms = terms_neutral, ctype = 1 }


return {

  nodes = {
    [1] = {  30, 1125, 50 },
    [2] = { 560, 1125, 50 },
    [3] = {  30,  925, 50 },
    [4] = { 560,  925, 50 },
    [5] = {  30,  685, 50 },
    [6] = { 360,  685, 50 },
    [7] = {  30,  465, 50 },
    [8] = { 560,  465, 50 },
    [9] = {  45,  130, 50 }
  },

  conduits = {
    -- Canaletas horizontais, 30x50 mm
    { f = 1, t = 2, s = 30*50, o = true },
    { f = 3, t = 4, s = 30*50, o = true },
    { f = 5, t = 6, s = 30*50, o = true },
    { f = 7, t = 8, s = 30*50, o = true },

    -- Canaletas verticais, 30x50 mm
    { f = 1, t = 3, s = 30*50, o = false },
    { f = 3, t = 5, s = 30*50, o = false },
    { f = 5, t = 7, s = 30*50, o = false },

    { f = 2, t = 4, s = 30*50, o = false },
    { f = 4, t = 8, s = 30*50, o = false },

    -- conduíte fictício - chicote até a barra terra
    { f = 7, t = 9, s = 50000, o = false }
  },

  cabletypes = {
    [1] = { name = "1,0 CZ", es = 2,  cost = 0.25 },    -- Comando
    [2] = { name = "2,5 PT", es = 6,  cost = 0.50 },    -- Força
```

```
    [3] = { name = "6,0 PT", es = 12, cost = 0.85 },
    [4] = { name = "2,5 VD/AM", es = 6, cost = 0.50 }   -- Terra
  },

  terminals = terms,
  conns = conns
}
```

Anexo II – Biblioteca GALILEO

```
-- Genetic Algorithm Library for Intelligent Electrical Optimization

module("galileo", package.seeall)

require "anyrandom"

TERM_MAX_FITNESS = 1
TERM_MAX_GENERATIONS = 2
TERM_MAX_STAGNATION = 3

-- Optimizations
local assert = assert
local min = math.min
local max = math.max

local mrandom = anyrandom.init()


-- Partially mapped cross over operator
local function pmx(a, b)
    assert(#a == #b, "Bad chromossomes")
    local ta, tb = mrandom(1, #a), mrandom(1, #b)
    local ia, ib = min(ta, tb), max(ta, tb)

    local ma, mb = { }, { }     -- used values
    for i = ia, ib do
        ma[b[i]] = true
        mb[a[i]] = true
    end

    local na, nb = { }, { }     -- children
    local ja, jb = 1, 1
    for i = 1, #a do
        if ia <= i and i <= ib then
            na[i] = b[i]
            nb[i] = a[i]
        else
            -- Find next unused values
            while ma[a[ja]] do
                ja = ja + 1
            end
            while mb[b[jb]] do
                jb = jb + 1
            end
            na[i] = a[ja]
            nb[i] = b[jb]
            ja = ja + 1
```

```
            jb = jb + 1
        end
    end

    return na, nb
end


-- Swap mutate operator
local function swmutate(a)
    local ia, ib = mrandom(1, #a), mrandom(1, #a)
    a[ia], a[ib] = a[ib], a[ia]
end



local function generate_population(top_seq, n_individuals)
    -- Create a basic chromossome
    local chrom = { }
    for i = 1, top_seq do
        chrom[i] = i
    end

    -- Shuffles the newly generated chromossome
    for i = 1, top_seq * 10 do
        swmutate(chrom)
    end

    -- Generate random individuals
    local chroms = { }
    for i = 1, n_individuals do
        chroms[i] = { }
        for j = 1, top_seq do
            chroms[i][j] = chrom[j]
        end
        for i = 1, top_seq do
            swmutate(chrom)
        end
    end

    return chroms
end

function new(elements, individuals)
    local o = {
        chroms = generate_population(elements, individuals),
        objfunc = nil,
        mprob = 0.02,
        cutpoint = 0.5,
        maxseq = elements,
        nindiv = individuals,
        maxfitv = nil,      -- Maximum fitness value
        maxgens = nil,      -- Maximum number of generations
```

```
        maxstag = nil       -- Maximum stagnated generations
    }
    setmetatable(o, { __index = _M })
    return o
end


function setObjFunc(self, f)
    assert(type(f) == "function", "Bad objective funcion")
    self.objfunc = f
end

function getObjFunc(self)
    return self.objfunc
end


function setMutationProb(self, mprob)
    assert(0 <= mprob and mprob <= 1, "Bad mutation probability")
    self.mprob = mprob
end

function getMutationProb(self)
    return self.mprob
end


function setCutPoint(self, cutpoint)
    assert(0 <= cutpoint and cutpoint <= 1, "Bad cuting point")
    self.cutpoint = cutpoint
end

function getCutPoint(self)
    return self.cutpoint
end


function setChromossome(self, chnum, chrom)
    assert(0 < chnum and chnum <= self.nindiv, "Bad chromossome number")
    assert(#chrom == self.maxseq, "Bad chromossome length")
    local usedvalues = { }
    local nchrom = { }
    for i = 1, #chrom do
        usedvalues[chrom[i]] = true
        ncrom[i] = chrom[i]
    end
    -- Ensure that all requiered values are set.
    for i = 1, self.maxseq do
        assert(usedvalues[i], "Bad chromossome data")
    end
    self.chroms[chnum] = nchrom
end
```

```
function getChromossome(self, i)
    return self.chroms[i]
end

function getChromossomes(self)
    return self.chroms
end


function setMaxFitness(self, max)
    self.maxfitv = max
end

function getMaxFitness(self)
    return self.maxfitv
end


function setMaxGenerations(self, max)
    assert(max > 0, "Bad number of generations")
    self.maxgens = max
end

function getMaxGenerations(self)
    return self.maxgens
end

function setMaxStagnation(self, max)
    assert(max > 0, "Bad number of generations")
    self.maxstag = max
end

function getMaxStagnation(self)
    return self.maxstag
end


local function compare_fitness(a, b)
    return a.fit > b.fit
end

function run(self)
    assert(self.objfunc, "No objective function defined.")
    assert(self.maxfitv or self.maxgens or self.maxstag,
        "No termination criteria defined.")

    -- Tables for execution data
    local chdata = { }
    for i = 1, #self.chroms do
        chdata[i] = { }
    end

    local gens = 0          -- Number of generations
```

```
local sgens = 0         -- number of stagnated generations
local lastbest = nil    -- Last best fitness

while true do

    -- Evaluate the chromossomes.
    for i, chrom in ipairs(self.chroms) do
        chdata[i].ch = chrom
        chdata[i].fit = self.objfunc(chrom)
        --print("chrom", i, chdata[i].fit, "-", unpack(chrom))
    end

    -- Sort chromossomes in descending order of fitness.
    table.sort(chdata, compare_fitness)

    -- Handle termination criteria

    if self.maxfitv and chdata[1].fit > self.maxfitv then
        return chdata[1].ch, chdata[1].fit, gens,
               TERM_MAX_FITNESS
    end

    gens = gens + 1
    if self.maxgens and gens > self.maxgens then
        return chdata[1].ch, chdata[1].fit, gens,
               TERM_MAX_GENERATIONS
    end

    if lastbest == nil or chdata[1].fit > lastbest then
        lastbest = chdata[1].fit
        sgens = 0
    else
        sgens = sgens + 1
    end

    if self.maxstag and sgens > self.maxstag then
        --print(chdata[1].fit, chdata[#chdata].fit)
        return chdata[1].ch, chdata[1].fit, gens,
               TERM_MAX_STAGNATION
    end

    -- Cut the population in two  using truncation selection.
    local cutp = math.floor(self.cutpoint * #chdata)
    assert(cutp > 2, "No individuals left")

    -- Replace discarted individuals with the breed of the
    -- selected ones.
    for i = cutp + 1, #chdata - 1, 2 do
        local ia, ib = mrandom(1, cutp), mrandom(1, cutp)
        chdata[i].ch, chdata[i+1].ch = pmx(chdata[ia].ch,
            chdata[ib].ch)
    end
```

```
        -- Mutate individuals and copy them back to the main object.
        for i, dt in ipairs(chdata) do
            if mrandom() < self.mprob then
                swmutate(dt.ch)
            end
            self.chroms[i] = dt.ch
        end

    end

    return nil -- error?
end
```