Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Dissertação de Mestrado

Desenho de Grafos: Uma Abordagem utilizando Programação Linear Inteira

Felipe Marques Terra

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

Orientador: Prof. Renato Cardoso Mesquita

Belo Horizonte, Agosto de 2009

Resumo

Um problema importante em visualização de dados é como organizar a informação a ser mostrada em estruturas que mostram entidades e as relações entre elas. Quando se tem um número extenso de entidades e de relações este problema pode ser muito complexo. Para um número grande de dados ou quando vários requisitos estéticos tem que ser obedecidos, o desenho automático de grafos se torna um problema relevante.

Este trabalho tem como objetivo o estudo de metodologias e o desenvolvimento de uma ferramenta computacional capaz de gerar desenhos de grafos de maneira automática, otimizando fatores como complexidade visual da informação representada no grafo, legibilidade, e diversos critérios estéticos. Para isso foram aplicadas técnicas de Programação Linear Inteira (PLI) em etapas de algoritmos de desenho de grafos, para que restrições específicas de desenhos de circuitos alimentadores de energia elétrica fossem contempladas de maneira mais eficiente.

Duas abordagens são apresentadas. A topologia-forma-métrica, que consiste em tratar o desenho em três etapas: planarização, responsável por definir um embutimento planar para o grafo; ortogonalização, responsável por definir ângulos de vértices e arestas; e compactação, responsável por obter coordenadas de um desenho o mais compacto possível. Outra abordagem, a hierárquica, visa dispor vértices em camadas, ou níveis, e possui resultados interessantes quando aplicada a determinados problemas de desenho de grafos, tais como esquemas que precisam realçar informações de direção de fluxos, como é o caso de alguns diagramas de sistemas de energia elétrica.

Desenhos de grafos gerados pelas diferentes abordagens são discutidos e comparados. Os resultados obtidos indicam que a abordagem PLI permite a inclusão de restrições de forma mais simples, possibilitando a inserção de mais requisitos estéticos nos desenhos automáticos.

Abstract

An important problem in data visualization is the information organization into structures that show entities and their relationships. This problem can be more complex as the number of entities increases. The way a graph is drawn can directly impact on the information quality and reliability. For large amount of data, or, when large number of aesthetic requirements had to be defined, the automatic graph drawing problem becomes a very relevant problem. The methodology study and the development of a computational tool designed to build automatic graph drawing are the objectives of this work, optimizing the graph data visual complexity, readability and many kinds of aesthetic requirements. To achieve these goals, the Integer Linear Programming (ILP) approach was used in algorithm steps, just for treating electric circuits drawing requirements in an efficient way. Two approaches are presented: the topology-metric-shape deals the drawing in three steps: planarization, responsible for the definition of a planar embedding; orthogonalization, responsible for vertices and edges angles definition; and the compaction, which defines the compact drawing coordinates. Another approach, the hierarchical one, targets a drawing in levels, and achieves better results in some kinds of graph drawing problems, like flow schemas in electric power systems diagrams. Graph drawings built with the different approaches are largely discussed and compared. The obtained results show that the ILP approach allows the insertion of requirements in the system in a simple way. This enables the insertion of many aesthetic requirements.

Agradecimentos

Primeiramente a Deus por simplesmente permitir que tudo isto esteja acontecendo e que eu tivesse saúde para conseguir trabalhar neste projeto.

À toda minha família por sempre me apoiar.

Ao meu orientador Renato Cardoso Mesquita e pela oportunidade que me foi dada e pelo auxílio valioso.

Aos amigos do GOPAC (Grupo de Otimização e Projeto Assistido por Computador) pela grande amizade e pelas ajudas freqüentes. Em especial: Adriano Lisboa, Alexandre Ramos, Dalmy Júnior, Diogo Batista, Douglas Vieira, Leonardo Mozelli, Luciano Pimenta, Marcelo Guedes, Marcos Flávio, Miguel Lima e Ricardo Adriano.

Aos alunos de iniciação científica Christiano Gouveia, Raphael Duarte, Thiago Campos e Vitor Baracho pela preciosa ajuda e dedicação.

Ao Vinícius Magalhães (Light/RJ), Daniel, Esdras, Arthur e Gisele (Concert Technologies) pela cooperação no projeto P&D.

À Ana Flávia pela compreensão, carinho e amor.

Aos meus amigos por entenderem meus longos períodos de ausência.

A todos que direta ou indiretamente contribuíram para a realização desse trabalho.

A todos vocês, muito obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\alpha(u, v)$	Parâmetro utilizado na designação do ângulo entre duas arestas de H.
$\beta(u, v)$	Parâmetro que indica o número de dobras de uma aresta em H.
η	Rede de fluxos de um grafo ortogonal planar.
η_{hor}	Rede de fluxo horizontal para compactação
η_{ver}	Rede de fluxo vertical para compactação
Γ	O desenho de um grafo.
λ, μ, χ	Limite inferior, a capacidade e o custo de um arco (u, v) em η.
$\sigma(x)$	Fluxo de uma entidade x (face, vértice).
θ	Custo de fluxo em uma rede.
ε	Ordem cíclica que define um embutimento planar.
A_h	Conjunto de arcos horizontais para um grafo restrito.
A_v	Conjunto de arcos verticais para um grafo restrito.
$c_{(v,w)}$	Ângulo entre a extremidade esquerda do eixo de balanço horizontal de v e a extremidade w da aresta (v, w) incidente a v.
$D(v)$	O conjunto de arestas com origem em um vértice V.
D_h	Grafo restrito com arcos horizontais.
D_h	Grafo restrito com arcos verticais.
E	Conjunto de arestas de um grafo.
E	Uma aresta qualquer de um grafo.
E_h	Conjunto de arestas horizontais do grafo.
E_v	Conjunto de arestas verticais do grafo.

G	Um grafo.
H	Representação ortogonal de um grafo.
$m_{(v,w)}$	$c_{(v,w)} \mod 4$.
$S_d(v)$	O conjunto de arestas a serem anexadas a um lado d do vértice v.
S_h	Conjunto de segmentos horizontais do grafo.
S_v	Conjunto de segmentos verticais do grafo.
V	Conjunto de vértices de um grafo.
f	Face de um grafo
p	Total de ângulos-vértice em um vértice
u, *v*, *w*	Vértices de um grafo.

Capítulo 1

Introdução

1.1 Motivação

Desenho de grafos endereça o problema de construir representações gráficas de grafos, redes e estruturas combinatoriais relacionadas. As representações geométricas dos grafos vêm sendo investigadas pela matemática há séculos, tanto para a visualização das estruturas quanto para desenvolver a intuição sobre as estruturas tratadas. A organização e visualização de informação através de estruturas que demonstram entidades e relações de adjacência é uma problema importante quando tais informações forem complexas ou numerosas. A forma como um grafo é visualizado pode ser extremamente importante (Battista et al., 1999), se analisarmos que a qualidade do entendimento, confiabilidade e eficiência em termos de quantidade de informação são dependentes do resultado da visualização.

A geração automática de desenhos de grafos tem várias aplicações. Exemplos incluem a Engenharia de Software (diagramas de fluxos de dados, grafos de chamadas de sub-rotinas, hierarquias de classes e diagramas de objetos em programas orientados a objeto), Bancos de Dados (diagramas entidade/relacionamento), Sistemas de Informação (diagramas organizacionais), Sistemas de Tempo Real (redes de Paterson (1977), diagramas de transição de estado), Sistemas de Suporte à Decisão (redes PERT, árvores de atividade), Projeto de circuitos integrados (diagramas de circuitos), Sistemas de Energia (diagramas das redes de transmissão e distribuição), Inteligência Artificial (diagramas de representação do conhecimento), etc. Devido à natureza combinatorial e geométrica dos problemas investigados e à ampla faixa de domínios de aplicação, a pesquisa em Desenho de Grafos tem sido conduzida em diversas áreas, incluindo a Matemática Discreta (teoria topológica de grafos, teoria geométrica de grafos), Algoritmos (algoritmos em grafos, estruturas de dados, geometria computacional, VLSI), Computação Gráfica (linguagens visuais, interfaces gráficas, visualização de software) e Engenharia (nas diversas aplicações do desenho de grafos, como, por exemplo, em sistemas de energia e no projeto de circuitos). Por este motivo, um grande número de publicações na área está disponível. Um tutorial de Battista et al. (1994) indica mais de 300 artigos sobre o assunto e, certamente, este número é atualmente muitas vezes

maior.

Especificamente, o problema proposto a ser resolvido por este trabalho se refere aos circuitos alimentadores da Light/RJ. Tais circuitos representam as informações de conectividade entre os elementos elétricos que compõem a transmissão de energia, desde o alimentador, até o receptor. Além disso, foi visto que para gerar os esquemáticos de circuitos alimentadores, uma série de regras, ou critérios de desenho deveriam ser seguidos.

Este trabalho procura focar em uma metodologia que trata de maneira mais intuitiva os fatores que definem o problema da visualização de grafos. Um estudo foi feito para se implementar um sistema que pudesse ser moldado conforme o problema a ser resolvido (tratando critérios específicos do problema). A capacidade de especialização da solução, amparada por uma metodologia genérica que oferece esta especialização de maneira mais intuitiva foi o objetivo principal deste trabalho.

1.2 Trabalhos relacionados

Um desenho de um grafo pode ser gerado por diferentes abordagens, as quais estabelecem relações de precedência variadas entre critérios estéticos. Cada contexto de aplicação dita quais são as características desejadas para o desenho. Diagramas ortogonais, por exemplo, são utilizados em uma ampla gama de aplicações. Usualmente, desenhos de grafos que correspondem a este tipo de diagrama também adotam a convenção grid, na qual as coordenadas de vértices e dobras de arestas são números inteiros. Muitos algoritmos de desenho que seguem esse modelo são conhecidos (Battista et al., 1994).

O *Giotto* é um algoritmo para desenho de grafos em grid mais eficiente em termos de vários critérios estéticos (Battista et al., 1994). Este algoritmo segue a metodologia *topologia-forma-métrica* (Tamassia, 1987) composta pelas etapas de planarização, responsável por definir um embutimento planar para o grafo; ortogonalização, responsável por definir ângulos de vértices e arestas; e compactação, responsável por obter coordenadas de um desenho o mais compacto possível. A vertente do algoritmo Giotto, utilizada em grafos com vértices que possuem n arestas de entrada e/ou saída (com n podendo ser maior que 4) é chamada *Kandinsky* (Fößmeier and Kaufmann, 1996, 1997). O algoritmo *Column* (T.Biedl and Kant, 1998) trabalha sob premissas de ser independente do processo de planarização, que pode gerar vários vértices fictícios ao grafo, os quais contribuem para o aumento de complexidade e dificuldade de implementação do desenho. A primeira fase do algoritmo é responsável por ordenar os vértices de um grafo biconectado a partir de uma fonte e um sumidouro. Em seguida, os vértices do grafo são adicionados em linhas consecutivas do grid e colunas são alocadas para suas arestas, em uma combinação ótima. O algoritmo *Pair* (Papakostas and Tollis, 1994) segue as mesmas premissas do Column com modificações de implementação que o torna mais simples embora menos eficiente. Melo (2007) desenvolveu uma biblioteca genérica para manipulação de grafos, contendo rotinas para desenho de grafos baseadas nestas abordagens.

A base da abordagem PLI para desenhos de grafo está nos algoritmos Giotto e Kandinsky. A grande premissa para se utilizar um sistema linear para etapas de ortogonalização e

compactação é poder ter simplicidade em se modelar as restrições de desenho. Como dito antes, isso é vital quando o contexto da aplicação dita quais critérios estéticos devem ser obedecidos em desenhos de grafos.

Desenhos de circuitos elétricos possuem características bem determinadas, como por exemplo, a idéia de fluxo de potência transferida entre os elementos do circuito. Algumas abordagens foram criadas para se evidenciar desenho de diagramas unifilares.

Rao and Deekshit (2003) apresentam, em seu trabalho, uma maneira de gerar diagramas unifilares com um método iterativo. Tal trabalho foi desenvolvido sob métricas de sistemas de circuitos alimentadores elétricos. Este trabalho concentra-se em estudar métodos de desenhos unifilares que privilegiam os critérios estéticos de agradabilidade e legibilidade. Entretanto, a preocupação com compactação do desenho (que poderia ser adquirida com um roteamento, resultado da ortogonalização no Giotto) é minimizada. Ong et al. (2000) mostram uma vertente do algoritmo de recolocação para gerar diagramas elétricos unifilares automaticamente em modo *grid*, não ortogonais. Neste trabalho, os autores mostram ganhos em eficiência ao utilizar estruturas especiais na abordagem de recolocação. Mas por se tratar de um algoritmo para desenho de diagramas unifilares, possui uma carência evidente, em termos de critérios estéticos mais abrangentes. Estes trabalhos mostram resultados para o problema de desenhos de circuitos elétricos, sob forma de diagramas unifilares. Além da necessidade da legibilidade, e da compactação do desenho, outros critérios estéticos fazem parte da gama necessária para se gerar bons esquemáticos de circuitos elétricos.

Sugiyama et al. (1981) mostram em seu trabalho uma abordagem eficiente, para desenho de hierarquias. Tal abordagem incorpora critérios importantes para desenho de circuitos elétricos e se mostra vantajosa, por simplicidade e eficiência, em comparação com outras similares.

1.3 Objetivos

Este trabalho tem como objetivo o desenvolvimento de uma ferramenta computacional capaz de gerar desenhos de grafos de maneira automática, otimizando fatores como complexidade visual da informação representada no grafo, legibilidade, e diversos critérios estéticos. Para isso foi proposto que seriam aplicadas técnicas de Programação Linear Inteira (PLI) em etapas de desenho dos grafos, para que restrições específicas em desenhos de grafos de circuitos alimentadores de energia elétrica fossem contempladas de maneira mais eficiente.

Foi implementada uma biblioteca em C++, onde foram escritos os métodos para geração de esquemáticos baseados em PLI. Tal biblioteca é usada no sistema de Geração Automática de Esquemáticos Ortogonais da Light cuja plataforma computacional para criação dos desenhos é fruto do presente trabalho de mestrado.

1.4 Organização do texto

O capítulo 2 apresenta diversos conceitos relacionados ao desenho de grafos, frisando as metodologias para desenho.

O capítulo 3 apresenta a metodologia de desenhos de grafos baseada em Programação Linear Inteira. Na etapa de ortogonalização, são definidos os ângulos entre arestas e dobras de arestas. Dada uma topologia, o passo de ortogonalização determina a forma de um desenho. Em uma representação ortogonal, vértices não possuem coordenadas e cada aresta (u,v) é equipada com uma lista de ângulos. Tal lista descreve as dobras que a representação ortogonal da aresta (u,v) terá no desenho final. Esta etapa faz parte da metodologia *Topologia-Forma-Métrica* detalhada na seção 2.3. Em seção específica, a ortogonalização será descrita em sua abordagem baseada em PLI. O capítulo ainda apresenta a etapa onde são geradas as coordenadas dos desenhos de grafos, definindo uma métrica para o esquemático. Esta etapa, a compactação, tem como entrada a saída da etapa de ortogonalização. Uma seção detalha a compactação em uma abordagem PLI onde são definidos tamanhos pré-determinados para os vértices, através de restrições inseridas no sistema.

O capítulo 4 apresenta a metodologia hierárquica para desenho de grafos. Tal metodologia dispõe os vértices e arestas em níveis e possui outros critérios estéticos e restrições para construção do esquemático. Nesta metodologia é destacado o processo de otimização do posicionamento de vértices que busca a minimização do número de cruzamentos de arestas num desenho.

O capítulo 5 mostra as implementações desenvolvidas para a solução de Geração Automática de Esquemáticos Ortogonais da empresa Light. Nesse capítulo são detalhados os componentes que configuram o sistema, sob o ponto de vista da obtenção dos dados através de informações dos circuitos alimentadores e da geração dos desenhos de tais circuitos. Os resultados das implementações são mostrados no capítulo 6.

Finalizando, o capítulo 7 apresenta algumas discussões sobre os métodos implementados e propostas de continuidade.

Capítulo 2

Desenho de Grafos

Nas diversas áreas onde se empregam rotinas de geração de desenho de grafos, existem requisitos que devem ser observados para que seja possível gerar bons desenhos, e, em cada área, tais requisitos podem ser bastante diferentes. Um desenho pode ser considerado ótimo em determinado contexto, mas pode não atender aos requisitos de alguma outra aplicação. O elemento essencial para uma metodologia de desenho de grafos é o ambiente particular no qual o desenho será utilizado. Na realidade, palavras como *domínio de aplicação* ou *ambiente onde será utilizado* são muito abstratas para serem utilizadas em algoritmos computacionais. Existem conceitos mais específicos para se descrever os requisitos de um desenho, a saber, as convenções de desenho, os critérios estéticos e as restrições (Battista et al., 1999). As próximas seções abordam, separadamente, cada um deles. Para entender os conceitos discutidos na seqüência, é importante estabelecer uma terminologia básica (Gross and Yellen, 1999; Harary, 2009; Bondy and Murty, 1976; Biggs et al., 1976; West, 2006; Wilson, 1996).

2.1 Grafos

Grafos são estruturas combinatoriais formadas por elementos simples que consistem em um conjunto de vértices e um conjunto de arestas. Vértices representam objetos concretos ou não, que possuem entendimento próprio e podem possuir propriedades adicionais como peso, cor, ou qualquer outro atributo que seja útil dentro do contexto da aplicação. As arestas representam as relações entre os vértices. Grafos podem representar modelos físicos como um circuito elétrico ou uma rede de computadores (Gross and Yellen, 1999; Harary, 2009; Bondy and Murty, 1976).

Definição 1 *Um grafo $G = (V; E)$ consiste na composição dos conjuntos finitos V contendo n_v vértices e E, contendo n_e arestas. O conjunto E é formado por pares não-ordenados de vértices $(u; w)$, onde $u,\ w\ \in\ V$.*

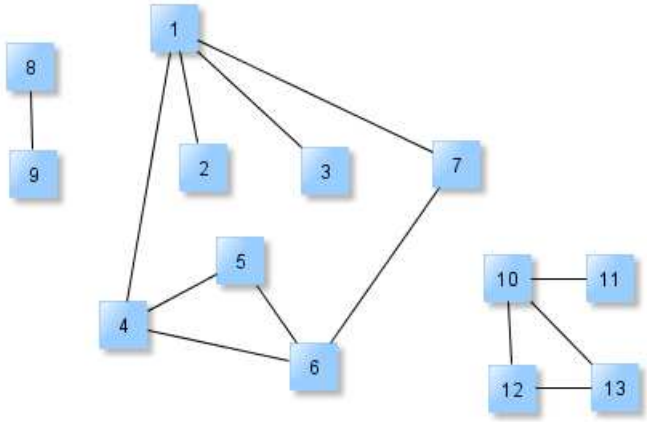


Figura 2.1: Um grafo com 13 vértices e 13 arestas.

Uma aresta $(u; w)$ com $u = w$ é chamada de auto-ciclo (*self-loop*). Se a aresta ocorre mais de uma vez em E, ela é uma aresta paralela. Um grafo simples é um grafo que não possui auto-ciclos ou arestas múltiplas. O grafo da figura 2.1 é simples. Os vértices finais ou extremidades de uma aresta $e = (u; w)$ são os vértices u e w. O vértice u é chamado de adjacente ao vértice w e a aresta e é chamada de incidente aos vértices u e w. No grafo da figura 2.1, os vértices 1 e 3 são adjacentes, porém os vértices 1 e 5 não são. O(s) vizinho(s) de um vértice w são os vértices adjacentes a w. No grafo da figura 2.1, os vizinhos de 1 são os vértices 2, 3, 4 e 7. O grau de um vértice w é o número de vizinhos que ele possui. No grafo da figura 2.1, o grau do vértice 1 é igual a 4.

Um grafo direcionado ou dígrafo é aquele onde as arestas são direcionadas, criando-se uma ordem para os vértices. No desenho de grafos direcionados, geralmente as arestas são representadas por setas, como na figura 2.2.

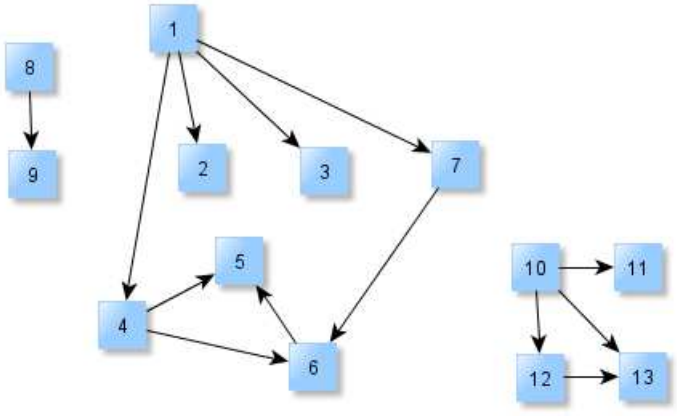


Figura 2.2: Exemplo de um dígrafo.

Um caminho (direcionado) em um grafo (direcionado) $G = (V; E)$ é uma seqüência $(v_1; v_2; ...; v_h)$ de vértices distintos de G, tal que $(v_i;\ v_{i+1}) \in E$ para $1 \leq i \leq h - 1$. No grafo da figura 2.1, a seqüência (1, 4, 5, 6) é um caminho e no grafo da figura 2.2,

a seqüência (1, 7, 6, 5) é um caminho direcionado. Um caminho (direcionado) é um ciclo (direcionado) se $(v_h;\ v_1) \in E$. Um grafo (direcionado) é acíclico se não possui ciclos (direcionados). No grafo da figura 2.1 o caminho (1, 4, 6, 7, 1) é um ciclo. O grafo direcionado da figura 2.2 não possui ciclos direcionados. Um desenho Γ de um grafo (dígrafo) G é uma função que mapeia cada vértice v a um ponto distinto $\Gamma(v)$ e cada aresta (u, v) a uma curva simples aberta $\Gamma(u, v)$, com pontos finais em $\Gamma(u)$ e $\Gamma(v)$. Um desenho de G é planar, se não houver interseção de arestas distintas. O grafo G é dito planar, se ele admite pelo menos um desenho planar. O grafo da figura 2.2 é planar.

Um desenho planar particiona o plano em regiões conectadas chamadas de faces. Uma face f é um laço de arestas, ou, uma seqüencia fechada de arestas. A face ilimitada é chamada de face externa, enquanto que as outras são chamadas de faces internas. Um embutimento planar de um grafo, é uma classe de equivalência de desenhos que se diferem pela ordem circular no sentido horário dos vizinhos de cada vértice. A figura 2.3 mostra um mesmo grafo apresentado com embutimentos planares diferentes.

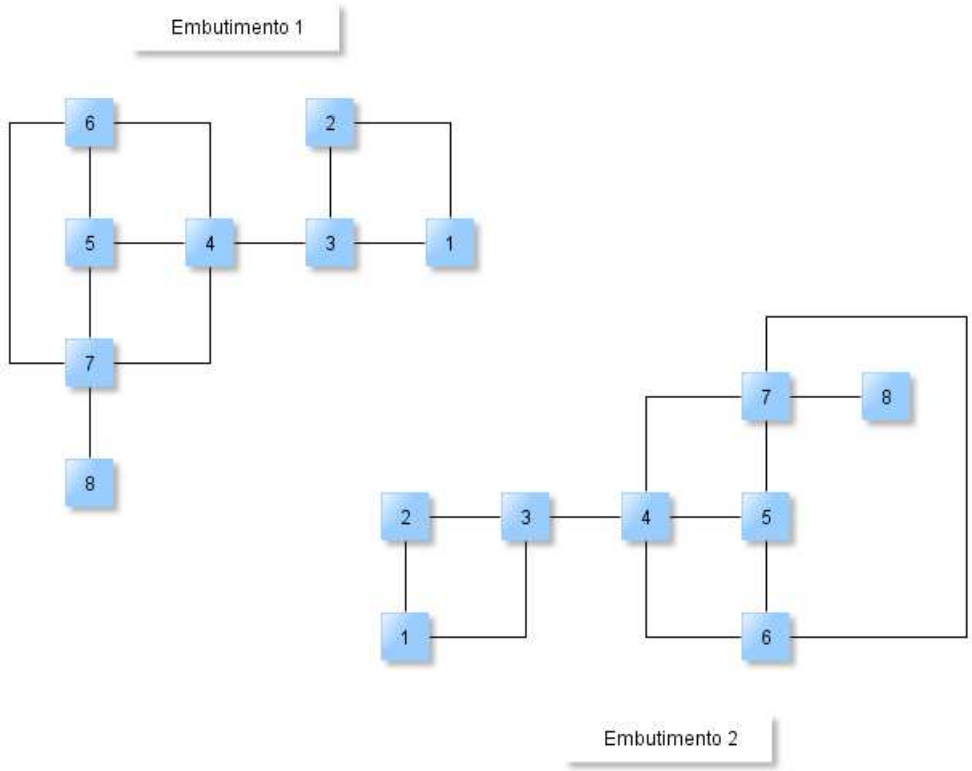


Figura 2.3: Dois desenhos de um mesmo grafo. Ambos embutimentos representam um mesmo grafo.

Um grafo é conectado se existe pelo menos um caminho entre u e v para qualquer par de (u, v) de vértices. A figura 2.1 apresenta um exemplo de grafo não conectado.

2.2 Paradigmas em Desenho de Grafos

Para desenhar um grafo G é importante levar em consideração as suas propriedades combinatoriais. O conhecimento sobre algumas propriedades predominantes de G (se é acíclico, direcionado, planar) pode direcionar qual tipo de algoritmo para desenho deve ser aplicado. Geralmente conhecemos a classe de grafos a qual G pertence. Essa classificação permite que determinados algoritmos próprios a determinadas classes de grafos destaquem melhor as propriedades combinatoriais do mesmo. Por exemplo, se G é um dígrafo acíclico, então pode ser importante desenhar todas as arestas seguindo a mesma direção, para enfatizar a inexistência de ciclos.

Assim, define-se que a classe do grafo em questão é um parâmetro de entrada para uma metodologia de desenho. Outros pontos de decisão também devem ser levados em consideração no desenho. Ítens como a percepção humana sobre as informações do desenho, ou mesmo, o ambiente em que será apresentado o desenho do grafo exercem grande dependência sobre os requisitos envolvidos na geração do desenho.

Nas seções que seguem, serão mostrados conceitos sobre convenção de desenhos, critérios estéticos e restrições, entre outros. Tais conceitos modelam a lista de requisitos necessários para que uma determinada classe de grafo (em conjunto com determinadas decisões de visualização) seja mais adequadamente descrita num desenho.

2.2.1 Convenções de Desenho

Uma convenção de desenho corresponde a uma regra básica que deve ser satisfeita pelo desenho. Por exemplo, para um desenho de diagrama de fluxo de dados relativo a uma aplicação de engenharia de software, podemos adotar a convenção de representar todos os vértices como caixas e todas as arestas como cadeias poligonais consistindo de segmentos horizontais e verticais. Uma convenção de desenho de um aplicativo real pode ser extremamente complexa e envolver vários detalhes do desenho. Algumas convenções bastante utilizadas estão descritas em seguida (Battista et al., 1999).

- **Poligonal** - Cada aresta do grafo é desenhada como uma cadeia poligonal.
- **Linhas-retas** - Cada aresta do grafo é desenhada com segmentos retos.
- **Ortogonal** - Cada aresta do grafo é desenhada como uma cadeia poligonal de segmentos horizontais e verticais. Curvas (de ângulo reto) entre segmentos de arestas são chamadas *dobras*.
- **Grid** - Vértices, cruzamentos de arestas e dobras de arestas possuem coordenadas inteiras.
- **Planar** - Convenção onde não há cruzamento de arestas no desenho.
- **Desenhos direcionados** - Um desenho de um grafo direcionado é chamado de *upward* quando todas as suas arestas possuem pelo menos um segmento voltado

para cima e nenhum voltado para baixo. Para o contrário (arestas direcionadas para baixo) é chamado de *downward*.

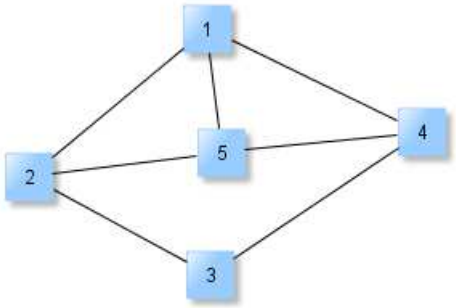


Figura 2.4: Grafo desenhado utilizando a convenção linhas-retas.

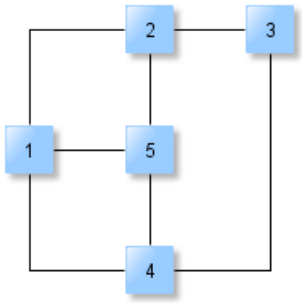


Figura 2.5: Grafo desenhado utilizando as convenções ortogonal, planar e grid.

Figura 2.6: Grafo utilizando a convenção de desenho direcionado (*downward*).

Linhas retas e desenhos ortogonais são casos especiais de desenho poligonal. Desenhos em linhas retas são comuns em livros e artigos sobre teoria dos grafos. Desenhos orto-

gonais são amplamente utilizados em circuitos esquemáticos e diagramas da engenharia de software. Desenhos planares são esteticamente atraentes, embora nem todos os grafos admitam tal desenho. Dígrafos acíclicos representando estruturas hierárquicas (ex.: diagramas de PERT, diagramas de herança de classes) são exemplos de desenhos direcionados ascendentes. Também é possível que um desenho adote mais de uma convenção. Desenhos de circuitos elétricos são normalmente executados utilizando as convenções Ortogonal, Grid e Planar.

2.2.2 Critérios Estéticos

Os Critérios estéticos especificam propriedades do desenho que devem ser aplicadas ao máximo possível, de maneira a se alcançar a facilidade de compreensão do desenho como um todo. Os seguintes critérios são comumente utilizados (Batine et al., 1985; Purchase and Cohen, 1995; Sugiyama et al., 1981):

- **Cruzamentos** - Minimização do número de cruzamentos entre arestas do grafo. O cenário ideal seria desenhos sempre planares, entretanto, nem todo grafo admite uma representação planar.
- **Área** - Minimização da área. Talvez um dos critérios mais importantes, pela necessidade de todo desenho caber em um local limitado. Este critério é especialmente importante na convenção grid, onde o tamanho das arestas é no mínimo uma unidade do grid (o desenho não pode ser reduzido ilimitadamente). Há duas formas de definir a área de um desenho: pelo fecho convexo (menor polígono convexo que engloba o desenho) ou pelo menor retângulo com lados horizontais e verticais cobrindo o desenho.
- **Comprimento total das arestas** - Minimização da soma do comprimento total das arestas. É um critério complementar à busca da minimização da área, assim como os outros critérios sobre comprimento de aresta.
- **Máximo comprimento das arestas** - Minimização do comprimento máximo de qualquer aresta.
- **Comprimento uniforme das arestas** - Minimização da variância no comprimento das arestas.
- **Número de dobras** - Minimização do número total de dobras nas arestas. Especialmente importante quando a convenção Ortogonal é utilizada. Por outro lado, não faz sentido se a convenção Linhas Retas é utilizada.
- **Máximo de dobras por aresta** - Minimização do número máximo de dobras em uma aresta.
- **Número de Dobras Uniforme** - Minimização da variância do número de dobras nas arestas.

- **Resolução Angular** - Maximização do menor ângulo entre duas arestas incidente no mesmo vértice. Especialmente importante quando a convenção linhas-retas é usada. Critério que atua diretamente na legibilidade das informações do desenho.
- **Razão de Aspecto** - Aproximação da razão de aspecto ótima do desenho. A razão de aspecto é definida como uma relação do comprimento da maior aresta e o comprimento da menor aresta do menor retângulo com cobertura (fecho convexo) horizontal e vertical. Importante em sistemas em que o desenho deve se adequar o mais perfeitamente possível às dimensões de um visualizador (ex.: sistemas sinóticos com visores de grande largura).
- **Simetria** - Enfatizar as simetrias do grafo. Existem modelos matemáticos que definem precisamente tais simetrias e seus respectivos desenhos (Melo, 2007; Battista et al., 1999; Lipton et al., 1985).

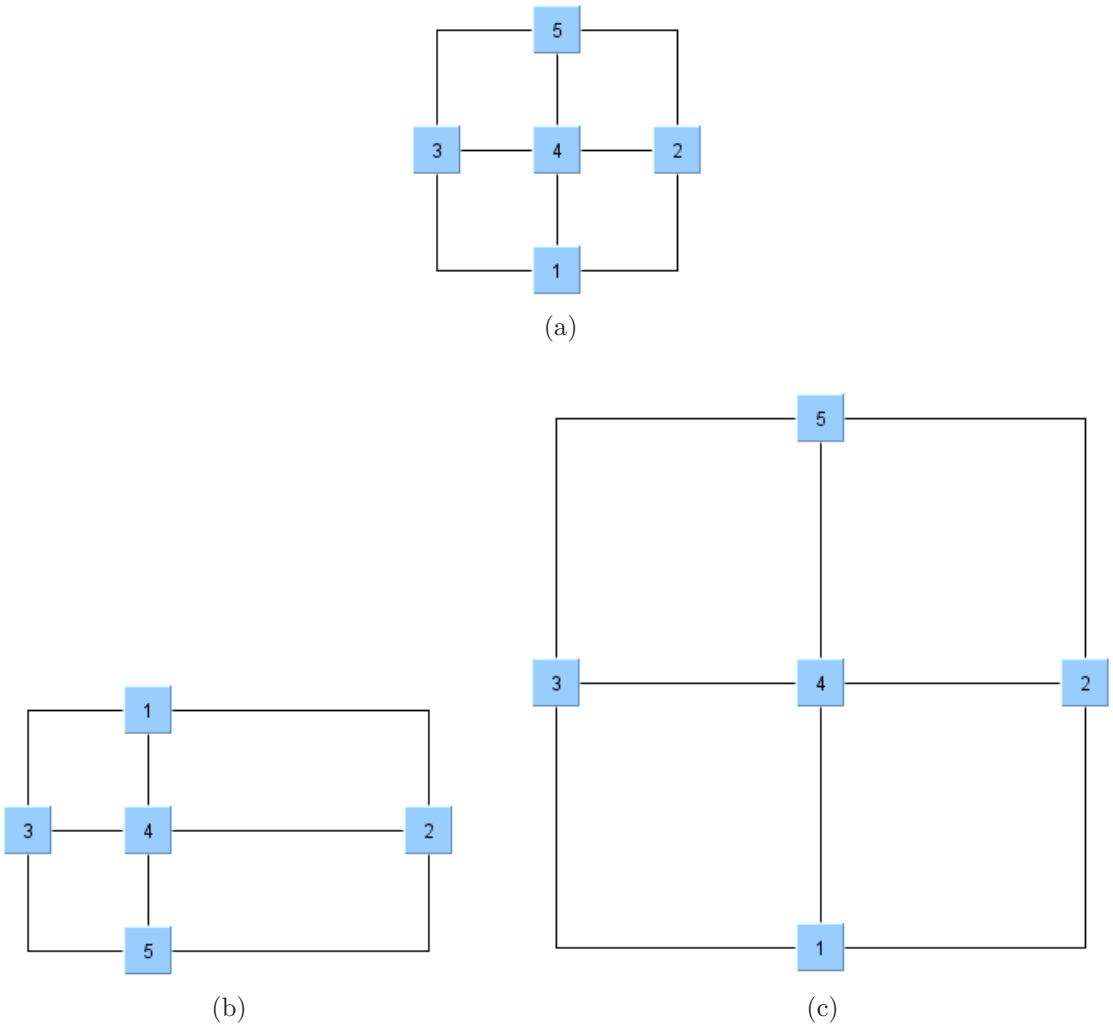
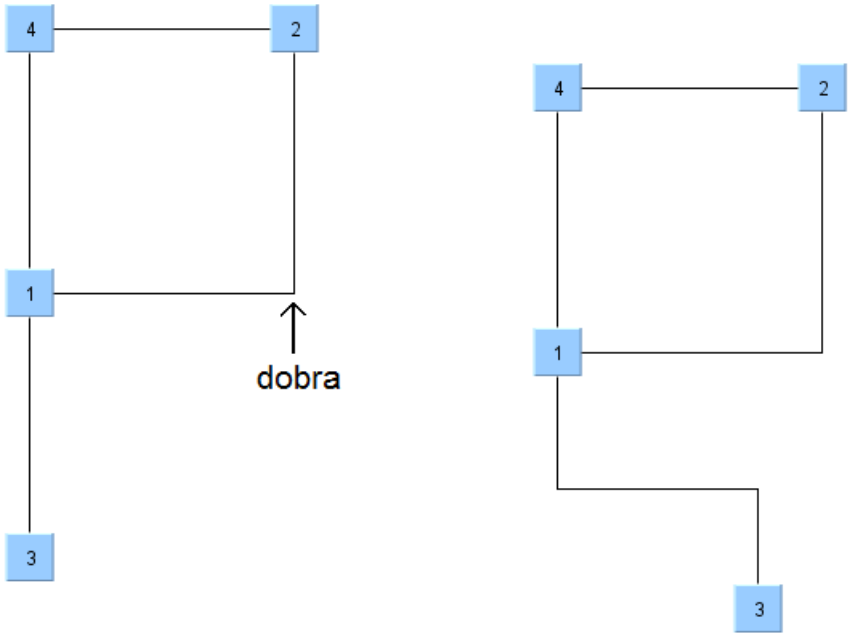


Figura 2.7: Exemplo de desenhos com áreas diferentes. A minimização de área busca atingir um desenho mais compacto possível.



(a) (b)

Figura 2.8: Exemplo de desenhos com número de dobras de arestas diferentes. A minimização de número de dobras de arestas busca oferecer um desenho mais legível (menos complexo).

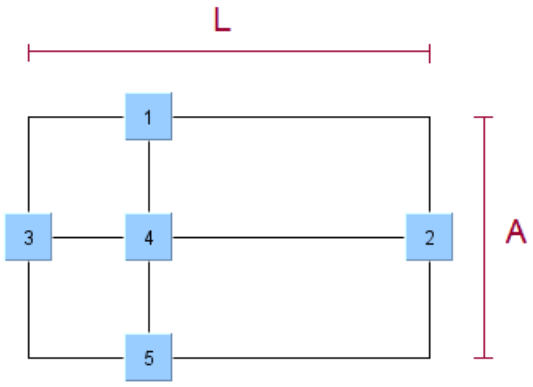


Figura 2.9: Controlar a razão de aspecto ou, a relação entre largura e altura de um desenho, permite adequar o grafo a uma tela sem que se tenha perdas de informação do desenho, por eventual mal posicionamento.

Os critérios citados são intuitivamente associados a problemas de otimização. O agrupamento de vários critérios, inclusive, remete a problemas de otimização multiobjetivo (Deb, 2003; Ehrgott, 2005). Por exemplo, o problema poderia ser: utilizando a convenção ortogonal, desenhe um grafo com área mínima, com número mínimo de mudanças de direção nas arestas, nenhuma interseção entre as arestas e atendendo a limites mínimos de resolução e razão de aspecto da área de tela disponível.

Entretanto, é comum a existência de conflitos entre critérios indicando que um sistema de otimização que tente contemplar todos simultaneamente pode não ser factível. Conseqüentemente, as metodologias de desenho de grafos estabelecem relações de precedência e adotam uma abordagem de divisão do processo em subetapas, nas quais cada algoritmo resolve um problema específico de forma independente, conforme veremos na subseção 2.2.5.

2.2.3 Restrições

Enquanto convenções de desenho e estética são regras e critérios gerais que se referem ao desenho do grafo como um todo, restrições se referem a subgrafos específicos e subdesenhos. As restrições mais utilizadas estão listadas abaixo (Kosak et al., 1994; Tamassia et al., 1988):

- **Centralização** - Posicionar um dado vértice no centro do desenho.
- **Externalização** - Posicionar um dado vértice no limite exterior do desenho.
- **Agrupamento** - Posicionar um conjunto de vértices próximos uns dos outros.
- **Seqüência (esquerda/direita e cima/baixo)** - Desenhar um dado caminho horizontalmente alinhado da esquerda para direita (verticalmente alinhado de cima para baixo).
- **Forma** - Desenhar determinado subgrafo em uma forma predefinida.

2.2.4 Eficiência dos algoritmos

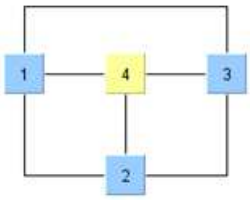
Este critério é extremamente importante quando tem-se que obter respostas em tempo real, por exemplo, em certos sistemas. Nesse caso, é comum a utilização de heurísticas que podem diminuir ordens de grandezas em determinados algoritmos (levando-se em consideração a separação de metodologias em sub-partes, que resolvem problemas distintos e independentes em desenho de grafos).

As convenções, critérios estéticos, o contexto da aplicação, a eficiência dos algoritmos e as restrições formam os parâmetros fundamentais associados às metodologias de desenho de grafos (Battista et al., 1999).

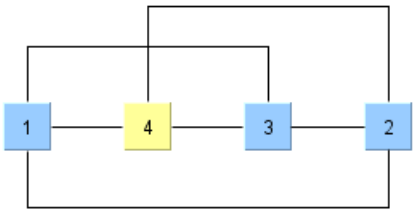
2.2.5 Precedência entre Critérios Estéticos

A maior parte das metodologias de desenho estão baseadas nas seguintes observações:

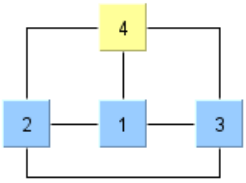
- Os critérios estéticos freqüentemente entram em conflito uns com os outros. Desse modo, é inevitável que se eleja um em detrimento de outro.



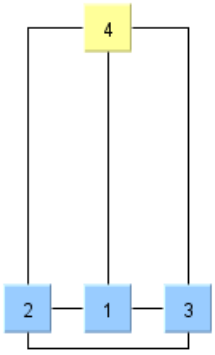
(a) Centralização



(b) Seqüencia



(c) Externalização



(d) Agrupamento

Figura 2.10: Exemplo de uso de restrições de forma no desenho. O posicionamento de um elemento ou um grupo de elementos pode mudar a forma do desenho, centralizando um dado grupo de elementos, externalizando, ou até mesmo colocando elementos em seqüencia.

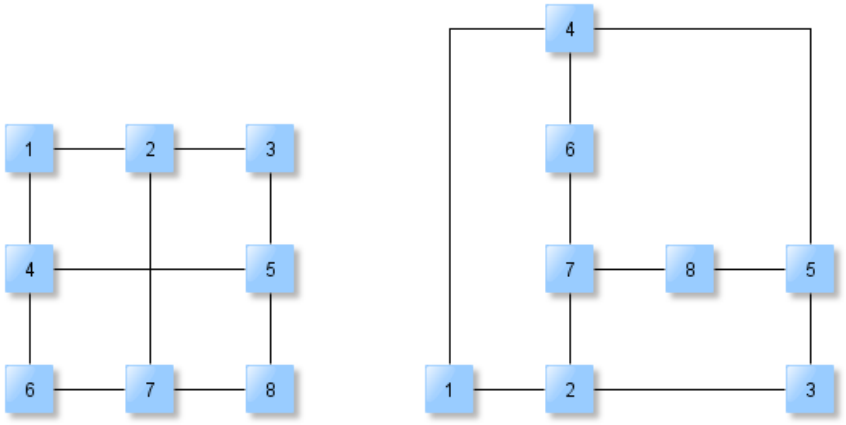
- Ainda que os critérios estéticos não entrem em conflito, é geralmente difícil levar em conta todos eles ao mesmo tempo.

Por exemplo, na figura 2.11 há dois desenhos em convenção grid ortogonal, um minimizando o número de dobras e outro minimizando o número de cruzamentos. Para este grafo, não existe um desenho em grid ortogonal que minimize ambos critérios estéticos simultaneamente.

É notado que a precedência de um critério sobre outro é diretamente ligado à adequação da aplicação. As estratégias apresentadas na literatura geralmente dividem o processo de desenho de grafos em uma seqüência de passos algorítmicos, cada um adequado para satisfazer uma certa subclasse de critérios estéticos. As metodologias mais populares são:

- **Topologia-Forma-Métrica**: (Tamassia, 1987; Tamassia et al., 1988; Batini et al., 1986). Esta estratégia foi projetada para construir desenhos ortogonais em grid, com tratamento homogêneo de uma grande variedade de critérios estéticos e restrições. Será melhor detalhada em seção próxima (2.3).

- **Hierárquica**: (Sugiyama et al., 1981; Carpano, 1980; Warfield, 1977). Esta metodologia foi projetada para construir desenhos de grafos orientados, exprimindo relações de dependência. Também será detalhada em seção deste texto (2.4).



(a) Minimização de Dobras (b) Minimização de Cruzamentos

Figura 2.11: Dois desenhos ortogonais do mesmo grafo

- **Visibilidade**: (Battista and Tamassia, 1988; Battista et al., 1992), é uma metodologia de propósito geral para o desenho de grafos que utilizam a convenção poligonal.

- **Aumento**: É outra metodologia de propósito geral, para o desenho de grafos na convenção poligonal. Porém é menos apropriada que as anteriores para a adição de restrições (Battista et al., 1999).

- **Direcionada por Força**: (Battista et al., 1999). Simula um sistema de forças definido no grafo de entrada e gera um desenho de "mínima energia". Interessante para grafos não direcionados na convenção linhas-retas.

- **Dividir para Conquistar**: Bastante aplicada no desenho de grafos que possam ser, recursivamente, divididos em subgrafos. As principais aplicações são o desenho de árvores (Reingold and Tilford, 1981; Walker, 1990) e dígrafos série-paralelos (Bertolazzi et al., 1994).

2.3 Topologia-Forma-Métrica

Uma característica marcante desta metodologia é que ela tem como resultado desenho adequado nas convenções planar, ortogonal e grid. Battista et al. (1994) apresenta alguns de muitos algoritmos que implementam esta metodologia. Um exemplo clássico de aplicação prática é a confecção de circuitos integrados, nos quais o diagrama elétrico deve ser desenhado sob restrições de uma área específica (Soukup, 1972).

Esta seção detalha um dos mais conhecidos e eficientes algoritmos para este conjunto de convenções. O algoritmo também introduz os critérios estéticos de minimizar o número de dobras e cruzamentos de arestas para representações ortogonais em grids. A base de entendimento deste algoritmo está em sua divisão de etapas: topologia, forma e métrica.

- **Topologia**: está relacionada à natureza do grafo, ou, o esquema (mapa) das ligações vértices/arestas. Dois desenhos ortogonais possuem a mesma topologia se um pode ser obtido a partir do outro apenas através de deformações contínuas, que não alteram as seqüências das arestas contornando as faces do desenho. Isso significa que a definição da topologia de um grafo é a definição de um dos seus embutimentos planares.

- **Forma**: trata-se da obtenção de ângulos e direções de dobras entre arestas de um grafo (complementando a topologia). Desenhos ortogonais possuem a mesma forma se eles têm a mesma topologia e um pode ser obtido a partir do outro modificando apenas o comprimento dos segmentos que representam as arestas do grafo, sem alterar os ângulos formados por elas.

- **Métrica**: caracterizada pela obtenção de coordenadas (tamanhos, posicionamento) do desenho. Ambos dois grafos possuem mesma métrica, se o que diferem entre si é somente operações de rotação e translação.

É possível perceber uma relação incremental, entre as propriedades descritas anteriormente. Isso significa que existe uma forte tendência de divisão em etapas, para algoritmos que implementam a metodologia (Batini et al., 1986). A saber, as etapas são:

1. *Planarização*: obtém um embutimento planar;

2. *Ortogonalização*: obtém informações de ângulos entre arestas e entre segmentos de arestas (dobras);

3. *Compactação*: obtém as coordenadas do desenho;

Cada uma destas etapas implementam os resultados esperados para cada uma das propriedades da metodologia *Topologia-Forma-Métrica*. Outros critérios estéticos e restrições podem ser introduzidos. Por exemplo, na etapa de planarização podemos forçar que certos vértices fiquem na área externa do desenho (objetivando um aumento de foco na visualização deste vértice, ou, determinando associações físicas, por exemplo, para o caso de vértices que representam entrada e saída de circuitos). Na etapa de ortogonalização podemos forçar que certas arestas não possuam dobras, ou impor seqüências específicas de dobras em algumas arestas. Na etapa de compactação, podemos forçar que certos vértices tenham coordenadas maiores ou menores que outros vértices.

Neste estudo, é mostrado que, de acordo com Eiglsperger et al. (2000), Eiglsperger and Kaufmann (2001), Eiglsperger et al. (2001), Klau and Mutzel (1999a), Klau and Mutzel (1999b) e Klau (2001) existem abordagens baseadas em Programação Linear Inteira (PLI) que tornam a inclusão de critérios e restrições de desenho uma atividade mais facilitada. Na realidade, tais abordagens são concebidas para melhor modelagem de sistemas com várias restrições. Nas sub-seções posteriores serão descritas as etapas da metodologia *Topologia-Forma-Métrica*, sob suas características básicas e originais. Nesse trabalho, também serão descritas as abordagens em PLI das etapas de ortogonalização e compactação.

2.3.1 Planarização

Uma das técnicas de planarização de grafos é construída a partir de sucessivas aplicações de uma operação de planarização, a qual possui como princípio básico a substituição de cruzamentos de arestas por vértices falsos. Um teste averiguando o critério de parada deste algoritmo é necessário: determina se um desenho já é planar. A fórmula de Euler (Bondy and Murty, 1976) garante que em um grafo simples planar, com V vértices, o número de arestas, E, máximo, é dado por $E \leq 3V - 6$. No entanto, isso não garante que grafos simples planares com número de arestas menor que $3V - 6$ sejam planares. Além disso, a planarização não apenas testa a planaridade do grafo, mas constrói um embutimento planar para o mesmo.

De forma sucinta, o objetivo principal da planarização é reduzir ao máximo o número de cruzamento de arestas. Este problema é equivalente ao problema do máximo subgrafo planar, bastante estudado por Jünger and Mutzel (1997), Jayakumar et al. (1986), Kant (1992) e Nardelli and Talamo (1984). Ambos são problemas *NP-hard*, o que leva grande parte dos algoritmos de planarização a utilizarem heurísticas.

Uma heurística bastante comum para computação de um máximo subgrafo planar é apresentada por Battista et al. (1999) e é definida por inserção de arestas, uma a uma do grafo original, verificando-se se cada inserção torna o grafo planar ou não. Nesse processo, para cada aresta (u, v), é preciso encontrar um caminho de custo mínimo no grafo dual do embutimento atual (que está sendo montado incrementalmente), e assim, inserir a aresta ao embutimento.

Dado um grafo planar, G, seu dual (geométrico), $G*$, pode ser construído de acordo com os passos apresentado por Wilson (1996). Existem vários algoritmos para cálculo do caminho de menor custo. Quando todas arestas do grafo possuírem o mesmo peso, uma busca em largura (BFS) pode ser utilizada. Para os outros casos, os algoritmos de Dijkstra (Dijkstra, 1959) e Bellman-Ford (Bellman, 1958) e (Ford and Fulkerson, 1962) são boas opções. O importante é que exista alguma forma de navegabilidade entre o grafo dual e o primal, já que o caminho mínimo é encontrado no primeiro, e a adição de arestas é feita no segundo.

O resultado das iterações de inserções das arestas é um embutimento planar (quando isto for possível).

2.3.2 Ortogonalização

A etapa de ortogonalização é responsável por criar uma *representação ortogonal* H, a partir de um embutimento planar resultante da planarização. O resultado desta etapa é a obtenção dos ângulos entre arestas que partem ou chegam em cada vértice, e os ângulos das dobras das arestas, que determinam sua direção.

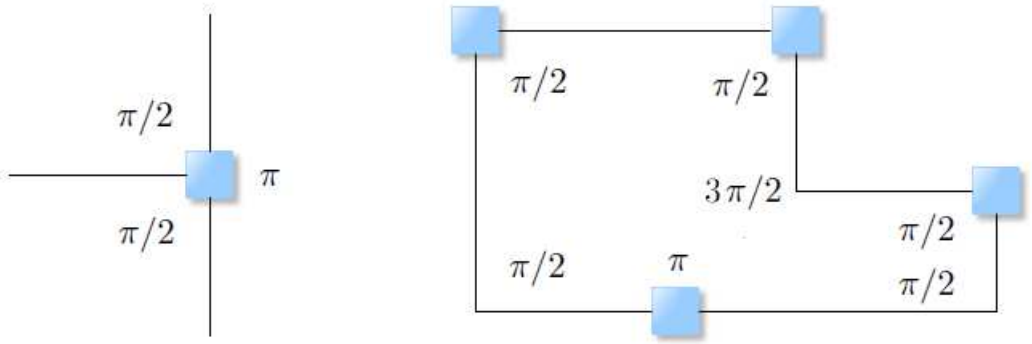
Este processo de minimização, de acordo com o método original mostrado por Battista et al. (1999), é descrito em um problema de fluxo em redes, onde os vértices são produtores de ângulos e as faces são consumidores dos mesmos. Todo fluxo gerado pelos vértices deve ser consumido pelas faces e cada unidade de fluxo corresponde a um

ângulo de $\pi/2$.

A seguir é descrito o método original de ortogonalização de embutimento planar baseado em fluxo de redes que minimiza o número de dobras de arestas. Este algoritmo possui custo quadrático e foi primeiramente apresentado por Tamassia (1987) com variações, refinamentos e extensões dadas em Fößmeier and Kaufmann (1996), Tamassia (1985) e Tamassia et al. (1988).

Sendo Γ um desenho ortogonal de um embutimento planar G', existem dois tipos de ângulos em Γ:

- Ângulo gerado por duas arestas incidentes em um vértice comum, chamado de *ângulo-vértice*.
- Ângulos formados por dobras de arestas (entre segmentos de aresta consecutivos), chamados *ângulos-aresta*.



(a) Ângulos-vértice (b) Ângulos-arestas

Figura 2.12: Ângulos em torno de um vértice e dentro de uma face em um desenho ortogonal planar

Isto nos leva às seguintes conclusões:

Lema 1 *Em um desenho planar ortogonal, a soma dos ângulos-vértice de cada vértice deve ser igual a* 2π.

Lema 2 *Para cada face interna* f *de* Γ, *a soma dos ângulos-vértice e de dobras é igual a* $\pi(p-2)$, *onde* p *é o total de ângulos. Para a face externa, a soma é igual a* $\pi(p+2)$.

A prova destes lemas é apresentada em Tamassia (1987).

Considere um embutimento planar de um grafo bidirecional G, com vértices de grau máximo quatro. O número total de ângulos-vértice dentro de uma face f é designado como $n_a(f)$. Como G é bidirecionado, para cada aresta $e = (u, v)$, também existe a aresta $e = (v, u)$. A aresta que percorre determinada face f no sentido anti-horário é chamada de aresta anti-horário. O conjunto de arestas com origem em um vértice V é denotado por $D(v)$, enquanto que o conjunto de arestas anti-horário da face f é denotado por $D(f)$.

Dado um desenho planar ortogonal de G, são definidos os seguintes valores para cada uma de suas arestas:

- $\alpha(u,v)\cdot\pi/2$ é o ângulo em um vértice u formado pelos primeiros segmentos da aresta (u,v) e a próxima aresta anti-horário ao redor de u.
- $\beta(u,v)$ é o número de dobras ao longo da aresta (u,v) com o ângulo de $\pi/2$ em seu lado esquerdo.

Os parâmetros α e β são definidos como os parâmetros que caracterizam uma representação ortogonal (desenhos com os mesmos valores destes parâmetros são considerados equivalentes). Portanto, uma representação ortogonal H é definida através da atribuição de valores inteiros α e β para todas arestas de G, de forma que as seguintes propriedades sejam satisfeitas:

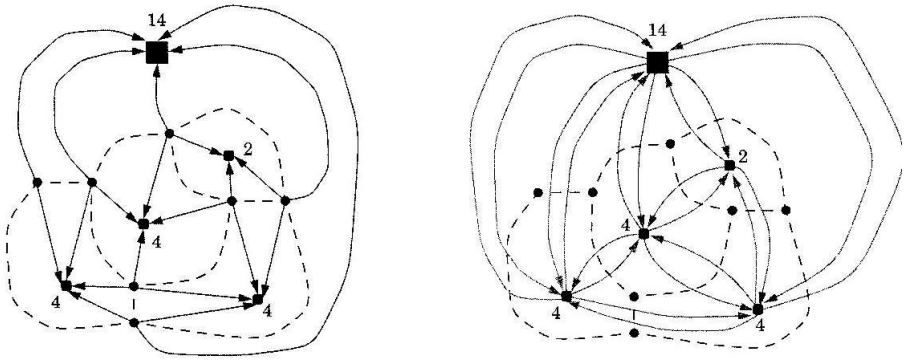
1. $1 \leq \ \alpha(u,v) \leq \ 4$
2. $\beta(u,v) \geq \ 0$
3. Para cada vértice u, a soma de $\alpha(u,v)$ para todas as arestas orientadas a partir de u é 4, ou:
$$\sum_{(u,v)\in\ D(f)} \alpha(u,v) = 4 \tag{2.1}$$
4. Para cada face interna *f*, a soma de $\alpha(u,v)+\beta(v,u)+\beta(u,v)$ para todas as arestas anti-horário de *f* é igual a $2n_a(f)-4$.
$$\sum_{(u,v)\in\ D(f)} \alpha(u,v)+\beta(v,u)+\beta(u,v) = 2n_a(f)-4 \tag{2.2}$$
5. Para a face externa, esta mesma soma é igual a $2n_a(f)+4$, ou:
$$\sum_{(u,v)\in\ D(f)} \alpha(u,v)+\beta(v,u)+\beta(u,v) = 2n_a(f)+4 \tag{2.3}$$

Tais ítens definem o sistema que deverá ser obedecido na minimização das dobras de arestas.

Transformação do problema em fluxo de redes

O problema de se encontrar um desenho ortogonal com o menor número de dobras é modelado através de uma rede de fluxo (Ahuja et al., 1993). Na rede equivalente os ângulos determinados pela ortogonalização representam o fluxo do modelo: produzidos nos vértices, transportados pelas faces, ao longo de suas dobras incidentes, e, em alguns casos, consumidos pela própria face. Como característica de um desenho ortogonal, os ângulos são múltiplos de $\pi/2$.

A premissa desta metodologia (em associar os ângulos ao fluxo) é de determinar um fluxo de custo mínimo, o que corresponde um desenho com menor número de dobras. A descrição abaixo explica como essa rede deve ser construída. Os valores de $\lambda(u, v)$, $\mu(u, v)$ e $\chi(u, v)$ indicam, respectivamente, o limite inferior, a capacidade e o custo de um arco (u, v), veja a figura 2.13:



(a) arcos de η de nodos-vértice para nodos-face.

(b) arcos de η entre nodos-face.

Figura 2.13: Rede η associada a um embutimento planar do grafo G (mostrado em linhas pontilhadas). Cada nodo-face f de η é mostrado com a quantidade de fluxo $\sigma(f)$ consumida: (a) arcos de η de nodos-vértice para nodos-face; (b) arcos de η entre nodos-face.

Os nós da rede η são os vértices e faces de G. Um nó derivado de um vértice é chamado nó-vértice. Um nó derivado de uma face é chamado nó-face. Um nó-vértice V de η produz fluxo $\sigma(v) = 4$, e, um nó-face f de η consome um fluxo $\sigma(v) = 2n_a(f) - 4$, se f é interna, ou $\sigma(v) = 2n_a(f) + 4$ para faces externas. Para cada aresta (u, v) de G, com faces f e G a sua esquerda e direita, respectivamente, η possui dois arcos:

- **Arco** (u, f): Este arco possui $\lambda(u; f) = 1$, $\mu(u; f) = 4$ e $\chi(u; f) = 0$. O arco $(u; f)$ representa a quantidade $\alpha(u, v)$. O limite inferior e a capacidade são colocados dessa forma em decorrência da definição de produção de fluxo por um vértice V. O custo associado a esse arco é zero, pois o ângulo se trata de um ângulo de vértice (note que o arco é do vértice u para face f) e a solução buscada é a de mínimo número de dobras (ângulos nas arestas).

- **Arco** (f, g): Este arco possui $\lambda(f, g) = 0$, $\mu(f, g) = \infty$ e $\chi(f, g) = 1$. O arco $(f; g)$ representa a quantidade $\beta(u, v)$. O limite inferior e a capacidade indicam que a aresta pode ou não conter dobras. A unidade do custo equivale à unidade de fluxo definida ($\pi/2$). Como ele ocorre em uma aresta (note que o arco é da face f para a face G), deve ser contabilizado.

O total de fluxo produzido pelos nós-vértices deve ser igual ao total de fluxo consumido pelos nós-faces. Portanto, a conservação do fluxo na rede η associada ao embutimento planar de G é descrita pela equação 2.4:

$$\sum_{v} \sigma(v) - \sum_{f} \sigma(f) = \sum_{v} 4 - \sum_{f} (2n_a(f) - 4) - 8 \tag{2.4}$$

O custo de um fluxo θ em uma rede construída conforme a descrição acima equivale ao número total de dobras de uma representação ortogonal H de G. Além disso, o fluxo de um arco entre nodos-vértices e nodos-faces, $\theta(u; f)$, corresponde ao valor $\alpha(u, v)$ da aresta associada em H, enquanto que o fluxo de um arco entre nodos-faces, $\theta(f; g)$, corresponde ao valor $\beta(u, v)$ dessa mesma aresta.

As figuras 2.14 e 2.15 mostram exemplos de montagem de uma rede de fluxo para um exemplo de desenho compactado.

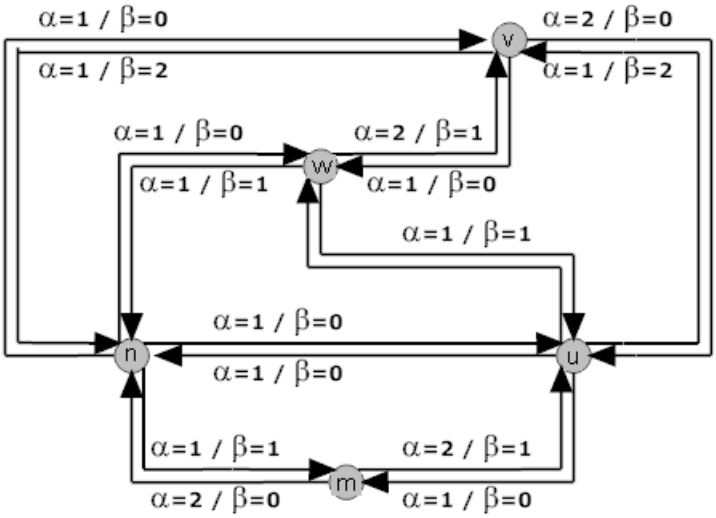


Figura 2.14: Exemplo de ortogonalização.

2.3.3 Compactação

A compactação é a etapa onde as coordenadas são atribuídas a uma representação ortogonal. Na compactação, os comprimentos das arestas são determinados seguindo um objetivo de construir um desenho de pequena área, utilizando apenas coordenadas inteiras para os pontos, já que a convenção grid é adotada.

A estratégia mostrada neste texto exige que as faces do grafo sejam retangulares. Todas as arestas $(u, v) \in D(f)$ devem possuir $\alpha(u, v) \leq 2$ e $\beta(v, u) = 0$. Na face externa, todas as arestas $(u, v) \in D(h)$ devem possuir $\alpha(u, v) \geq 2$ e $\beta(u, v) = 0$. Casos onde as faces não são retangulares, devem ser tratados refinando-se representações ortogonais com tais faces não retangulares afim de se obter faces sempre retangulares. Isso é conseguido inserindo-se vértices, ou arestas falsas.

O problema da compactação também pode ser resolvido através de abordagem de fluxo de redes. Neste caso, duas redes são criadas (η_{hor} e η_{ver}) , unindo-se pontos interiores de cada face da representação ortogonal, e seguindo as duas direções de minimização de

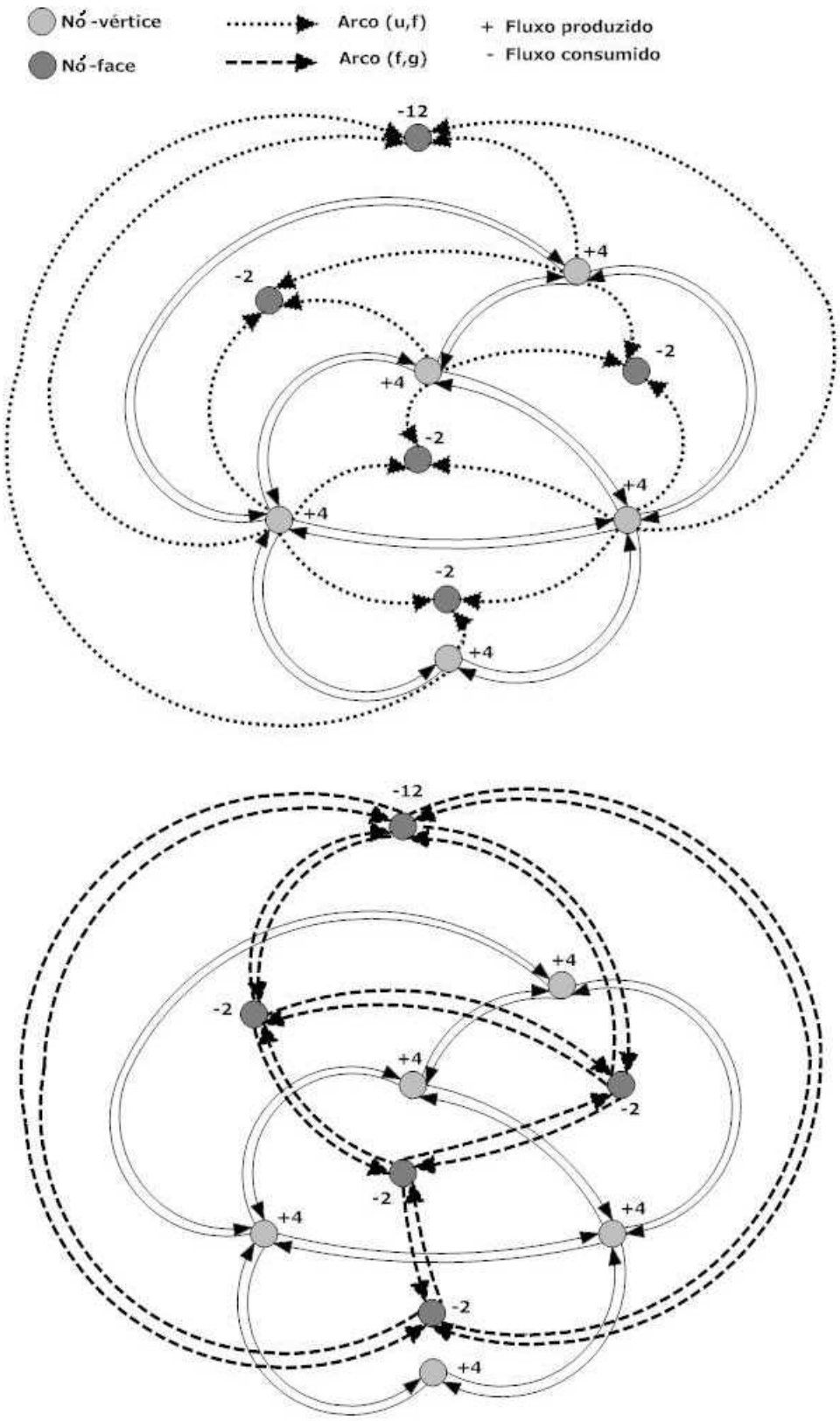


Figura 2.15: Rede associada ao embutimento planar da figura 2.14.

tamanhos: horizontal e vertical. A figura 2.16 mostra a distribuição dos pontos para as redes de fluxo citadas.

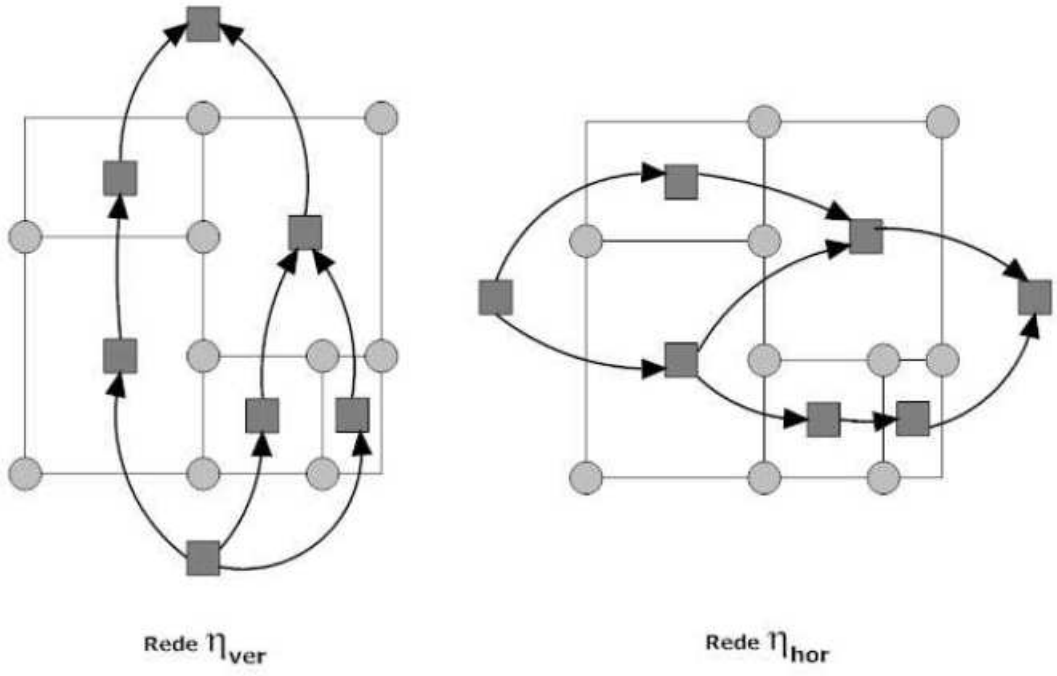


Figura 2.16: Redes η_{ver} e η_{hor} para configuração do sistema de compactação das arestas.

Estes sistemas de fluxos de rede horizontal e vertical são responsáveis por definir os tamanhos das arestas dispostas nas duas direções, e possuem como restrição básica a dimensão unitária para uma aresta sem dobras ligando dois vértices. Portanto, o problema é encontrar o fluxo de menor custo, em uma rede com limites inferiores nos arcos.

Complexidade da Compactação e Ortogonalização por Fluxo de Redes

A complexidade de tempo das etapas de ortogonalização e compactação da abordagem apresentada aqui é determinada pelos algoritmos de fluxo em redes. Os algoritmos de fluxo máximo por incremento de caminhos e fluxo máximo por formação de pré-fluxo possuem complexidade de tempo diferentes e fortemente dependentes da densidade do grafo (Sedgewick, 2002). De qualquer forma, nenhum deles garante complexidade de tempo inferior a $O(V^2)$. Segundo Foulds (1992) é possível obter um algoritmo mais elaborado chegando em uma complexidade de $O(V^{7/4}logV)$.

2.4 Metodologia Hierárquica

A metodologia hierárquica definida em Battista and Tamassia (1988), Sugiyama et al. (1981), Sugiyama (2002) e Eiglsperger et al. (2005) é empregada geralmente em dígrafos,

onde se deve enfatizar a informação das camadas hierárquicas do desenho. A metodologia hierárquica também é tratada em passos:

1. *Distribuição em Camada*: nesta etapa, primeiramente, são distribuídas as camadas (através de critérios específicos de requisitos, ou, automaticamente, via busca por largura). Cada aresta liga uma camada à sua posterior. Para o caso em que uma aresta liga dois vértices de camadas distantes, falsos vértices são criados em níveis das camadas intermediárias.

2. *Redução de Cruzamentos*: nesta etapa, o resultado da etapa de Distribuição em camadas é recebido como entrada. Para reduzir os cruzamentos, a ordem dos vértices em cada nível é otimizada, mirando-se um resultado de minimização de cruzamentos global. Neste caso, um sistema linear pode ser usado para tal minimização.

3. *Atribuição de Coordenadas x*: nesta metodologia, cada nível definido na hierarquia possui elementos com mesma coordenada *y*. Nesta etapa de atribuição de coordenadas *x* são produzidas coordenadas finais para os vértices preservando a ordenação e a redução de cruzamentos obtidos no passo anterior. No final dos passos, o desenho final é obtido pela primeira representação de cada aresta com um segmento de linha reta e então ocorre a remoção dos vértices "fictícios". (Desta maneira, arestas compridas podem ser representadas por linhas poligonais).

A abordagem hierárquica suporta alguns tipos de restrição. Por exemplo, dois dados vértices na mesma camada podem ser forçados a estarem próximos um do outro no passo de redução de cruzamentos. Assim, vértices em diferentes camadas podem estar verticalmente alinhados durante o passo de atribuição à coordenada *x*. No capítulo 4 é detalhado um dos algoritmos mais importantes na metodologia hierárquica, a abordagem de Sugiyama et al. (1981).

Capítulo 3

Desenhos de Grafos baseados em PLI

A estratégia mais efetiva para tratar desenhos ortogonais é a que usa os conceitos apresentados anteriormente, de topologia, forma e métrica. A implementação mais eficiente de tal estratégia, segundo Eiglsperger et al. (2000) é o GIOTTO (Tamassia, 1987; Tamassia et al., 1988; Battista et al., 1995). Nesta estratégia, como visto antes, são efetuadas três etapas para formação do desenho, a Planarização, a Ortogonalização e a Compactação. Tidas como críticas para a abordagem, as etapas de Ortogonalização e Compactação são problemas resolvidos através de sistemas de fluxo de redes, como problemas de minimização iterativos.

A versão original do algoritmo GIOTTO atendia apenas a desenho de grafos de grau máximo igual a 4. Na abordagem Kandinsky (Fößmeier and Kaufmann, 1996, 1997) o conceito GIOTTO foi estendido de maneira que grafos com grau maior que 4 pudessem ser desenhados sem perda do controle do número final de vértices.

Neste capítulo é apresentado um método para ortogonalização considerando uma larga variedade de restrições, onde se tem uma maior flexibilidade no desenho de grafos de acordo com os requisitos dos usuários (Eiglsperger et al., 2000). A abordagem aqui mostrada é baseada nos algoritmos GIOTTO e Kandinsky.

3.1 Modelagem por Programação Linear Inteira

Em um modelo de programação linear inteira (Papadimitriou and Steiglitz, 1982; Chvatal, 1983), existe a necessidade de se otimizar (maximizar ou minimizar) uma função objetivo da forma:

$$f(x_1, x_2, ..., x_3) \;=\; a_1x_1 \;+\; a_2x_2 \;+\; ... \;+ a_nx_n \tag{3.1}$$

sujeita a restrições modeladas por equações e inequações lineares, tais como:

$$\begin{array}{rcl} A[x] & \leq & b \\ x & \geq & 0 \end{array} \tag{3.2}$$

Nas equações 3.1 e 3.2, x é o vetor das variáveis, a matriz A é chamada matriz de coeficientes e o vetor b vetor de restrições. Se as variáveis do sistema são inteiras, diz-se que a programação linear é inteira. O objetivo de um problema de PLI como o mostrado acima é minimizar ou maximizar a função objetivo, determinando o valor de cada variável.

A grande vantagem deste tipo de modelagem no desenho de grafos está no fato de que qualquer novo critério estético a ser aplicado ao desenho exige apenas a inclusão de novas inequações (que modelam uma restrição) na matriz de coeficientes, sem necessidade de alterações no algoritmo em si. Esta característica torna as metodologias baseadas em PLI bastante atraentes tendo em vista a flexibilidade oferecida por estas para a inclusão de novas restrições.

3.2 Ortogonalização em Programação Linear Inteira

A etapa de ortogonalização PLI (Eiglsperger et al., 2000; Eiglsperger and Kaufmann, 2001), da mesma forma que a baseada em fluxo, recebe como entrada do sistema um embutimento planar, resultante da etapa de Planarização, de um grafo $G = (V, E)$, com vértices V e arestas E, dados pela ordem cíclica em sentido horário das arestas incidentes de cada vértice v. $\varepsilon(v)$ é a seqüencia cíclica de arestas atrelada a cada vértice v. Cada aresta $e = (v, w)$ pertencente a E aparece duas vezes em ε: uma vez como (v, w) em $\varepsilon(v)$ e outra como (w, v) em $\varepsilon(w)$. Complementando o conjunto de dados apresentado, define-se como F o conjunto de faces do grafo. Seja F_{in} o conjunto de faces internas, e F_{out} o conjunto de faces externas, cada face f é armazenada como uma lista de arestas ordenadas. Cada aresta e em uma face f é direcionada tal que a face prescrita f esteja à direita de e (figura 3.1).

Para modelar um sistema PLI equivalente ao problema de fluxo de redes da ortogonalização, a definição de dobras de arestas deve ser implementada, pois, é exatamente este o ponto de minimização de todo o sistema de ortogonalização. Para cada aresta (u, v) pertencente a ε, há uma variável $r_{(u,v)}$, que conta o número de dobras de ângulo de $\pi/2$ (dobras à direita, seguindo o sentido definido para ε) e uma variável $l_{(u,v)}$, que conta o número de dobras de $3\pi/2$ (dobras à esquerda), ambas ao longo da aresta na direção $u - v$. As variáveis $r_{(u,v)}$ e $l_{(u,v)}$ contam o número de dobras da direção contrária $(v - u)$ (uma dobra de $3\pi/2$ em uma direção é uma dobra de $\pi/2$ na direção contrária) (figura 3.2). Para se obter um desenho com número mínimo de dobras, minimiza-se a soma das variáveis de dobras:

$$\sum_{(u,v) \in E} (r_{(u,v)} + l_{(u,v)}) \tag{3.3}$$

Nesta modelagem é necessário, ainda, que seja modelado um outro tipo de variável, que definirá as direções de saída de arestas em cada vértice. A variável $a_{(u,v)}$ define um valor de ângulo entre $e = (u, v)$ e seu predecessor cíclico em $\varepsilon(v)$, no vértice v. O valor r de uma variável $a_{(u,v)}$ corresponde a um valor de $r \cdot \pi/2$ do ângulo correspondente.

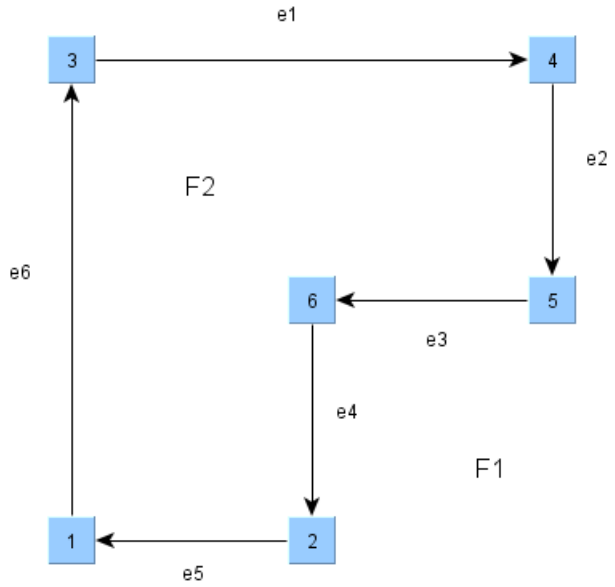


Figura 3.1: Definição de faces utilizada em Eiglsperger et al. (2000). Nesta figura há duas faces: $F1 : \{e1, e6, e5, e4, e3, e2\}$ e $F2 : \{e1, e2, e3, e4, e5, e6\}$.

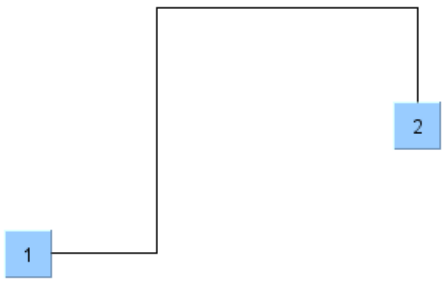


Figura 3.2: Percorrendo-se $e(1, 2)$, tem-se duas dobras à direita ($r_{(1,2)} = 2$) e uma à esquerda ($l_{(1,2)} = 1$). Na direção contrária, $e(2, 1)$, tem-se $r_{(2,1)} = 1$ e $l_{(2,1)} = 2$.

Todos os ângulos incidentes têm que ser múltiplos de $\pi/2$, de forma que se deve demandar, explicitamente, que as variáveis possuam valores inteiros.

Isto leva ao seguinte sistema de minimização inteira (Eiglsperger et al., 2000):

$$
\begin{array}{l}
\min \quad \sum_{(v,w)\in\ E}(r_{(v,w)} + l_{(v,w)}) \quad \text{sujeito a} \\
(T1) \quad \sum_{(v,w)\in\ \varepsilon(v)} a_{(v,w)} = 4 \quad \forall\ v\ \in\ V \\
(T2) \quad \sum_{(v,w)\in\ f}(a_{(v,w)} + l_{(v,w)} - r_{(v,w)}) \ = \ \left\{ \begin{array}{ll} 2k-4 & f \in\ F_{in},\ |f| = k \\ 2k+4 & f \in\ F_{out},\ |f| = k \end{array} \right. \quad \forall\ f\ \in\ F \\
(T3) \quad l_{(v,w)} \ = \ r_{(w,v)} \quad \forall\ (v, w)\ \in\ \varepsilon \\
l_{(v,w)},\ r_{(v,w)}\ \in\ \mathcal{N} \quad \forall\ (v, w)\ \in\ \varepsilon \\
a_{(v,w)}\ \in\ 1, ..., 4 \quad \forall\ (v, w)\ \in\ \varepsilon
\end{array}
\tag{3.4}
$$

Em Tamassia (1987), mostra-se que as condições acima são suficientes para gerar desenhos ortogonais.

3.2.1 Kandinsky

O algoritmo Kandinsky pode tratar vértices com grau superior a quatro, permitindo ângulo zero entre arestas incidentes a um dos quatro lados de um vértice. Neste modelo, tem-se um grid de linhas mais espessas, pois atribui-se a cada uma delas um conjunto de linhas mais finas onde as arestas são roteadas. Isto significa que os vértices, neste modelo, têm largura e altura.

Para assegurar um desenho correto, a propriedade *bend-or-end* deve ser mantida, o que implica no controle do tamanho dos vértices. Existe também uma preocupação com os casos em que arestas sairão num mesmo lado de um vértice: nesse caso, há uma grande possibilidade destas arestas se cruzarem, caso um controle de saída não seja efetuado. Tal controle pode ser efetuado via dobras de vértice, um conceito aderido à modelagem Kandinsky e garante por meio de ângulos de saída de arestas que a interseção para estes casos seja evitada.

Cada aresta pode ter duas dobras de vértice, uma para cada vértice que nela incide. Para este propósito, introduz-se para cada aresta (v,w) pertencente a ε quatro variáveis: $lb^v_{(v,w)}$, $rb^v_{(v,w)}$, $lb^w_{(v,w)}$ e $rb^w_{(v,w)}$, que representam as dobras de vértice nos vértices v e w. As variáveis $lb_{(v,w)}$ e $rb_{(v,w)}$ que denotam as dobras de face (como as definidas na modelagem anterior) na aresta (v,w). Para simplificar, escreve-se $l_{(v,w)} = lb^v_{(v,w)} + lb_{(v,w)} + lb^w_{(v,w)}$ e $r_{(v,w)} = rb^v_{(v,w)} + rb_{(v,w)} + rb^w_{(v,w)}$. Comparando a notação entre as formulações GIOTTO e Kandinsky, observa-se que $l_{(v,w)}$ é sempre o número total de dobras à esquerda de (v,w). A formulação Kandinsky completa é dada pelo conjunto de equações 3.5:

$$
\begin{array}{ll}
\min & \sum_{(v,w)\in\ E}(r_{(v,w)}\ +\ l_{(v,w)}) \quad \text{sujeito a} \\
(K1) & \sum_{(v,w)\in\ \varepsilon(v)} a_{(v,w)} = 4 \quad \forall\ v\ \in\ V \\
(K2) & \sum_{(v,w)\in\ f}(a_{(v,w)} + l_{(v,w)} - r_{(v,w)})\ =\ \left\{ \begin{array}{ll} 2k-4 & f\in\ F_{in},\ |f|=k \\ 2k+4 & f\in\ F_{out},\ |f|=k \end{array} \right. \quad \forall\ f\in\ F \\
(K3) & lb^v_{(v,w)} + rb^v_{(v,w)}\ \leq\ 1 \quad \forall\ (v,w)\ \in\ \varepsilon \\
(K4) & a_{(u,v)} + lb^v_{(v,w)} + rb^v_{(v,w)}\ \geq\ 1 \quad \forall\ (v,w),(v,u) \text{ subseqüente em } \varepsilon(v) \\
(K5) & \left\{ \begin{array}{lcl} lb^v_{(v,w)} & = & rb^v_{(w,v)} \\ lb_{(v,w)} & = & rb_{(w,v)} \quad \forall\ (v,w)\ \in\ \varepsilon \\ lb^w_{(v,w)} & = & rb^w_{(w,v)} \end{array} \right. \\
lb^v_{(v,w)},\ rb^v_{(v,w)}, lb^w_{(v,w)},\ rb^w_{(v,w)}\ \in\ \mathcal{N} \quad \forall\ (v,w)\ \in\ \varepsilon \\
lb_{(v,w)},\ rb_{(v,w)}\ \in\ \mathcal{N} \quad \forall\ (v,w)\ \in\ \varepsilon \\
a_{(v,w)}\ \in\ 0,...,4 \quad \forall\ (v,w)\ \in\ \varepsilon
\end{array}
\tag{3.5}
$$

A figura 3.3 mostra um exemplo de desenho ortogonal aplicando-se o Kandinsky.

Apesar da equivalência entre as abordagens baseadas em redes de fluxo e aquelas baseadas em PLI, a extensão da abordagem GIOTTO para o modelo Kandinsky, utilizando

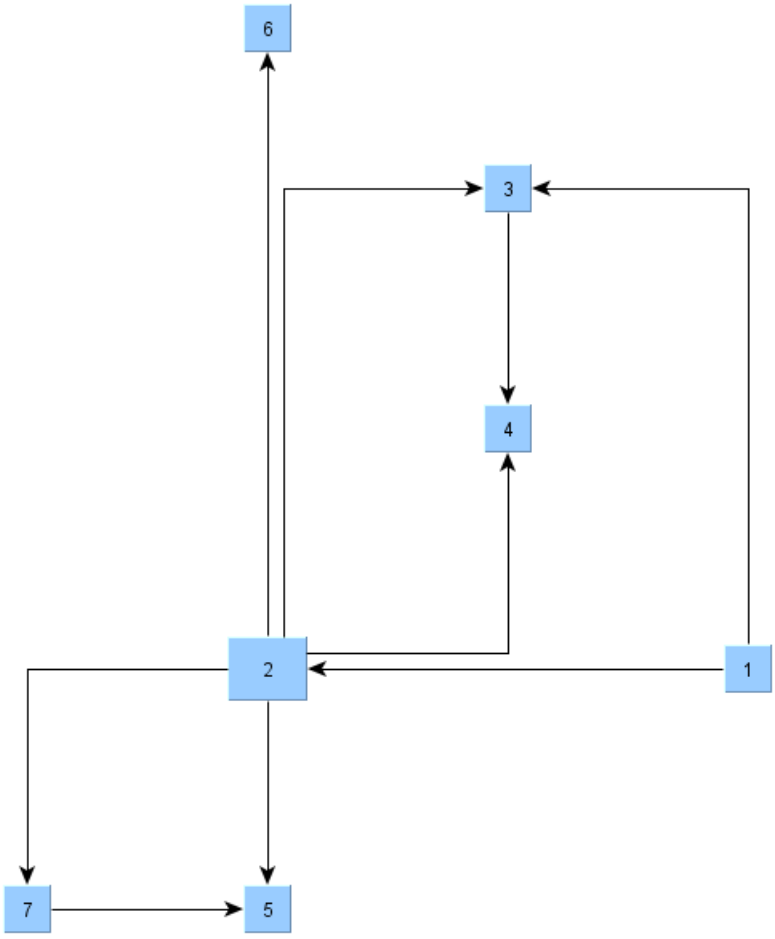


Figura 3.3: Um exemplo para um desenho com base em Kandinsky.

redes de fluxo é bastante complicada. A configuração da PLI pode ser formulada de uma maneira muito mais elegante e emprega maior facilidade na inserção de restrições variadas de desenho. A seção a seguir detalha a modelagem de algumas restrições para desenhos ortogonais, bastante importantes no âmbito de desenho de circuitos elétricos/eletrônicos.

3.2.2 Restrições Especiais em Ortogonalização

Existem alguns tipos de requisitos de desenho de grafos, que permitem o importante objetivo de se ler grandes quantidades de informações em grandes grafos, numa área pequena. Requisitos básicos como os que compõem a formulação descrita anteriormente objetivam minimização de área, de número de cruzamentos, entre outros. Outros tipos de requisitos poderiam ser:

- Definição das direções das arestas (*upward*, *rightward*).
- Definição de locais certos onde as arestas sairão em um vértice (restrições de posicionamento de saída).

- Definição de tamanho de vértices.

Como este trabalho objetiva a implementação de um sistema de quadros sinóticos automático para os circuitos alimentadores da empresa Light/RJ, alguns critérios específicos deste sistema (listados com detalhes no capítulo referente ao Sistema Light) tiveram que ser modelados. Tais critérios puderam descrever bem alguns elementos especiais dos circuitos alimentadores (tais como algumas chaves seccionadoras utilizadas na representação da Light).

Como dito anteriormente, a modelagem de critérios estéticos em desenho de grafos dentro da abordagem de programação linear inteira se dá através da inserção de equações e inequações ao modelo previamente estabelecido.

Mostra-se, nesta seção, um conjunto de restrições encontradas na literatura (Eiglsperger et al., 2000) para a modelagem de critérios estéticos (restrições). Algumas destas modelagens foram utilizadas no sistema de geração de sinóticos implementado neste trabalho de mestrado.

Direcionamento de arestas

Eiglsperger et al. (2000) apresenta um modelo de restrições capaz de direcionar grafos em um desenho. A referência chama tais restrições como *restrições de aresta*. Como definido anteriormente, um desenho *upward* é um desenho onde todas as arestas verticais sempre apontam para cima. Como requisito para um desenho *upward*, é preciso que o embutimento planar seja um embutimento *upward* (Fößmeier and Kaufmann, 1994; Garg and Tamassia, 1994). Este é um embutimento onde as arestas incidentes de um vértice v qualquer são somente ordenadas em sentido horário, e também possuem a primeira aresta de chegada (se não possuir arestas de chegada, a primeira aresta de saída) marcada como uma aresta $ancora(v)$ (Eiglsperger et al., 2000). Uma aresta $ancora$ é uma aresta definida como inicial na seqüencia cíclica ε.

A idéia básica para este algoritmo é introduzir um *eixo de balanceamento* para cada vértice. Tal eixo de balanceamento divide o vértice em dois retângulos. Para um desenho *upward*, por exemplo, o eixo se define horizontalmente, e, todas as arestas de saída dos vértices devem estar conectadas no retângulo de cima, e as arestas que chegam nos vértices, devem estar conectadas no retângulo de baixo. A figura 3.4 mostra a disposição dos eixos de balanceamento num trecho de desenho *upward*.

Para complementar esta idéia, e definindo as regras para este grupo de restrições, é preciso ressaltar que todos os eixos de balanceamento dos vértices que compõem o grafo devem ser orientados na mesma direção. Além disso, as dobras das arestas não podem criar segmentos de arestas que ferem o direcionamento do desenho (por exemplo, nos desenhos *upward* não são permitidos segmentos de arestas apontando para baixo).

Para gerar as restrições que caracterizam as condições escritas, é preciso definir, para cada aresta $e = (v, w) \in \varepsilon(v)$; c_e, denotando o ângulo entre a extremidade esquerda do eixo de balanceamento de v e o final da aresta e, incidente a v. Tal variável é definida restringindo-se o seu intervalo de valores entre -2 e 0, para arestas de entrada, e entre 0

e 2, para arestas de saída. Dessa forma, fica intuitivo perceber que as arestas de saída se posicionam por cima do eixo. Também é possível perceber que, para uma aresta e' seguindo e em $\varepsilon(v)$, $c_{e'} = c_e + a_{e'}$. Desta equação, tem-se que c_e pode ser expresso como $c_e = \sum_{e' \in A(e)} a_{e'} + c_v$, onde $A(e) \subseteq \varepsilon(v)$ é o conjunto de arestas entre $ancora(v)$ e $e = (v, w)$ em $\varepsilon(v)$, incluindo e, e, a variável c_v denota o ângulo entre $ancora(v)$ e a extremidade esquerda do eixo de balanço. Para assegurar que todas as arestas incidam no lado correto do vértice, as condições seguintes têm que ser inseridas na formulação PLI:

$$\begin{aligned} -2 \leq c_{(v,w)} \leq 0 \quad & \forall\, v \in V, \forall\, (v,w) \in \varepsilon_i(v) \\ 0 \leq c_{(v,w)} \leq 2 \quad & \forall\, v \in V, \forall\, (v,w) \in \varepsilon_o(v) \end{aligned}$$

Para estas restrições, $\varepsilon_i(v)$ é o subconjunto de $\varepsilon(v)$ composto das arestas de entrada. $\varepsilon_o(v)$ é o subconjunto análogo, para arestas de saída.

Entretanto, ainda são necessárias algumas condições adicionais, que garantirão que os eixos de balanceamento dos vértices do grafo estejam em direções sempre paralelas. Para isso, a restrição seguinte é suficiente (Eiglsperger et al., 2000):

$$c_{(v,w)} - l_{(v,w)} + r_{(v,w)} - c_{(w,v)} = 2 \quad \forall\, (v,w) \in \varepsilon_o(v) \tag{3.6}$$

Este critério é fundamental para construção de desenhos que devem mostrar sentido de fluxo (como é o caso dos desenhos dos circuitos elétricos alimentadores da Light). Veremos, mais adiante, que a inserção deste tipo de restrição ao sistema impactaria bastante negativamente o número de soluções factíveis para os desenhos. Após vários testes confrontando o resultado visual e legibilidade, foi decidido que uma abordagem hierárquica, ortogonal, seria mais eficiente e simples de se implementar, sob ponto de vista de controle das possíveis soluções não factíveis que poderiam surgir ao se inserir outras restrições junto desta.

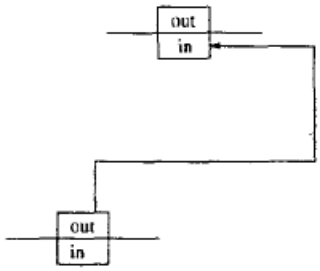


Figura 3.4: Trecho de desenho *upward* com destacamento do eixo de balanceamento. As partes *in* e *out* correspondem obrigatoriamente a partes de entrada e saída de arestas, respectivamente.

Posicionamento de arestas em um dos lados do vértice

A condição de posicionamento de arestas em um lado de um vértice qualquer é particularmente útil no caso da representação dos diagramas unifilares de distribuição, pois, tal característica consegue assimilar o sentido de distribuição de potência elétrica de maneira muito mais intuitiva.

Eiglsperger et al. (2000) apresentou um método baseado em PLI para a modelagem deste critério estético. Para tal, inicialmente devem ser feitas algumas definições. Seja $S_d(v)$ o conjunto de arestas a serem anexadas a um lado d do vértice v, onde d pode ser qualquer dos lados do vértice (lado esquerdo, lado direito, lado de cima ou lado de baixo). Insere-se, então, no conjunto de equações e inequações que definem o PLI, a variável $s^d_{(v,w)} \in \{0,1\}$. A variável se igualará a 1 se e somente se a aresta (v,w) tiver uma de suas extremidades no lado d, previamente definido. Assim, pelo fato de cada aresta ser incidente a exatamente um lado de um vértice, a equação 3.7 deve ser inserida no PLI.

$$\sum_{d \in \{esq,dir,cima,baixo\}} s^d_{(v,w)} = 1 \quad \forall (v,w) \in \varepsilon \tag{3.7}$$

Sejam:

- $c_{(v,w)}$ o ângulo entre a extremidade esquerda do eixo de balanço horizontal de v e a extremidade w da aresta (v,w) incidente a v;
- $m_{(v,w)}$ o resultado da operação $c_{(v,w)}\ mod\ 4$. Esta variável é usada para que os valores de $c_{(v,w)}$ sempre estejam no intervalo inteiro de 1 a 4;

Tem-se que o lado no qual uma aresta (v,w) é anexada ao vértice v é determinado pelo ângulo entre a extremidade esquerda do eixo de balanço horizontal do vértice v e a aresta (v,w). Tal condição se expressa nas equações a seguir:

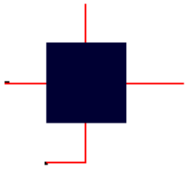
$$\begin{array}{rcl}
4m^{esq}_{(v,w)} + c_{(v,w)} & \leq & 3(1 - s^{esq}_{(v,w)}) \\
4m^{esq}_{(v,w)} + c_{(v,w)} & \geq & 0 \\
4m^{sup}_{(v,w)} + c_{(v,w)} & \leq & 1 + 3(1 - s^{sup}_{(v,w)}) \\
4m^{sup}_{(v,w)} + c_{(v,w)} & \geq & 1 \\
4m^{dir}_{(v,w)} + c_{(v,w)} & \leq & 2 + 3(1 - s^{dir}_{(v,w)}) \\
4m^{dir}_{(v,w)} + c_{(v,w)} & \geq & 2 \\
4m^{inf}_{(v,w)} + c_{(v,w)} & \leq & 3 + 3(1 - s^{inf}_{(v,w)}) \\
4m^{inf}_{(v,w)} + c_{(v,w)} & \geq & 3
\end{array} \tag{3.8}$$

Finalmente, para especificar que uma aresta (v,w) é incidente ao lado d de um vértice v, adiciona-se a restrição:

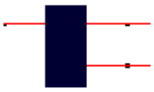
$$s^d_{(v,w)} = 1 \tag{3.9}$$

Este critério é bastante importante para caracterização de certos elementos de circuitos elétricos alimentadores da Light. A figura 3.5 mostra a caracterização de elementos conseguida ao se aplicar esta restrição no sistema que controla a ortogonalização dos desenhos

dos sinóticos de tais circuitos. No capítulo 5 os requisitos para os elementos especiais serão descritos em detalhes.



(a) Elemento especial: Chave seccionadora de 3 posições. Obrigatoriamente, cada arestas adjacente deve sair por um dos lados (mantendo um ângulo de 90 graus entre elas).



(b) Elemento especial: Chave seccionadora a gás. Obrigatoriamente, uma das arestas consideradas fonte deve estar sempre no lado oposto às outras, que permanecem no mesmo lado.

Figura 3.5: Caracterização dos elementos especiais conseguida através do controle de posicionamento de arestas.

Controle de tamanho mínimo de vértice pelas arestas

A abordagem *topologia-forma-métrica*, apesar de seu comprovado sucesso em diversas aplicações, ainda possui algumas desvantagens e problemas não resolvidos. Um dos problemas mais complicados é a dificuldade em se produzir desenhos com vértices de tamanho determinado pelo usuário. Atualmente, muitos algoritmos baseados nesta abordagem consideram os vértices como pontos infinitesimais (Tamassia, 1987), ou assumem que os vértices possuem tamanhos iguais (Batini et al., 1986; Fößmeier and Kaufmann, 1996).

Há, no entanto, uma forte demanda por aplicações que possibilitem a construção de desenhos ortogonais com vértices de tamanho arbitrário, definidos pelo usuário. É o que se observa, por exemplo, em representações que utilizam grafos para construção de diagramas UML, onde os elementos possuem distintas formas e tamanhos (Eiglsperger et al., 2003).

Eiglsperger et al. (2000) apresenta um conjunto de restrições a serem adicionadas à fase de ortogonalização do algoritmo Kandinsky que permitem especificar o tamanho mínimo de um vértice em dimensões $k_1 \times k_2$.

Para tal, deve-se inicialmente definir o lado d (lado esquerdo, lado direito, lado de cima, lado de baixo) de um vértice v e a restrição de tamanho k que deve ser aplicada a este lado. Assim, se d corresponde aos lados de cima ou de baixo, tem-se $k = k_1$; caso correspondam à esquerda ou à direita do vértice, utiliza-se $k = k_2$. Além disso, define-se E' como as $k-1$ arestas precedentes a e em $\varepsilon(v)$, incluindo e.

Toma-se cada intervalo $E' \subseteq \varepsilon(v)$ de tamanho k, onde a primeira aresta deixa o vértice v no lado d. Neste caso, assegura-se que a aresta é adjacente a um ângulo diferente de

0. E para assegurar a incidência desta aresta no lado d do vértice v, utiliza-se a seguinte restrição:

$$s^d_{(u,v)} + \sum_{(v,w)\in\ E'} a_{(v,w)} \geq 1 \forall\ E' \subseteq\ \varepsilon(v), |E'| = k+1, e = (u,v) \text{ primeira aresta em E'} \tag{3.10}$$

Na equação 3.10, a variável $a_{(v,w)}$ denota o ângulo entre a aresta (v,w) e seu predecessor cíclico no embutimento planar e incidente no vértice w. A figura 3.6 mostra um exemplo de controle de tamanho de vértice por arestas.

Neste trabalho, viu-se que utilizando restrições para tamanhos de vértices pré-determinados no sistema de compactação, obtém-se uma forma de controle de tamanho de vértices mais simples. Na seção 3.3, será mostrado como isso pode ser conseguido.

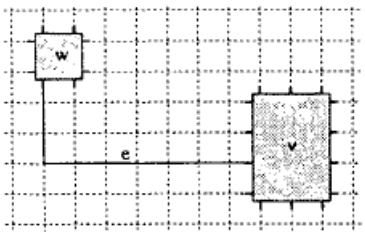


Figura 3.6: Uma aresta $e = (v,w)$ que inicia no vértice v na terceira unidade do lado esquerdo e finaliza no vértice w na segunda unidade do lado de baixo. O controle das entradas das arestas nas unidades de lado dos vértices força o controle de seu tamanho mínimo

Sistema Kandinsky Completo

Neste trabalho, foram utilizadas algumas restrições especiais, descritas nas seções anteriores. A equação 3.11 mostra a montagem completa do sistema linear inteiro da ortogonalização. A análise sobre a factibilidade do sistema pode ser estendida às análises feitas por cada restrição, no trabalho de Eiglsperger et al. (2000). O sistema montado por completo oferece uma região de solução mais restrita (obviamente percebido por causa das inserções das restrições). Entretanto, como veremos nos resultados, devido às características predominantes dos circuitos alimentadores da Light, que tendem a seguir como uma árvore, com aparente direcionamento de fluxo de potência da raiz para as pontas, sempre teremos uma solução com esta abordagem.

$$
\begin{array}{ll}
\min & \sum_{(v,w)\in\ E}(r_{(v,w)}\ +\ l_{(v,w)}) \quad \text{sujeito a} \\
(K1) & \sum_{(v,w)\in\ \varepsilon(v)} a_{(v,w)} = 4 \quad \forall\ v\ \in\ V \\
(K2) & \sum_{(v,w)\in\ f}(a_{(v,w)} + l_{(v,w)} - r_{(v,w)})\ =\ \begin{cases} 2k-4 & f\in\ F_{in},\ |f|=k \\ 2k+4 & f\in\ F_{out},\ |f|=k \end{cases} \quad \forall\ f\in\ F \\
(K3) & lb^{v}_{(v,w)} + rb^{v}_{(v,w)}\ \leq\ 1 \quad \forall\ (v,w)\ \in\ \varepsilon \\
(K4) & a_{(u,v)} + lb^{v}_{(v,w)} + rb^{v}_{(v,w)}\ \geq\ 1 \quad \forall\ (v,w),(v,u) \text{ subseqüente em } \varepsilon(v) \\
(K5) & \begin{cases} lb^{v}_{(v,w)} = rb^{v}_{(w,v)} \\ lb_{(v,w)} = rb_{(w,v)} \quad \forall\ (v,w)\ \in\ \varepsilon \\ lb^{w}_{(v,w)} = rb^{w}_{(w,v)} \end{cases} \\
(K6) & \begin{cases} 4m^{esq}_{(v,w)} + c_{(v,w)} \leq 3(1-s^{esq}_{(v,w)}) \\ 4m^{esq}_{(v,w)} + c_{(v,w)} \geq 0 \\ 4m^{sup}_{(v,w)} + c_{(v,w)} \leq 1+3(1-s^{sup}_{(v,w)}) \\ 4m^{sup}_{(v,w)} + c_{(v,w)} \geq 1 \\ 4m^{dir}_{(v,w)} + c_{(v,w)} \leq 2+3(1-s^{dir}_{(v,w)}) \\ 4m^{dir}_{(v,w)} + c_{(v,w)} \geq 2 \\ 4m^{inf}_{(v,w)} + c_{(v,w)} \leq 3+3(1-s^{inf}_{(v,w)}) \\ 4m^{inf}_{(v,w)} + c_{(v,w)} \geq 3 \end{cases} \\
(K7) & \sum_{d\in\{esq,dir,cima,baixo\}} s^{d}_{(v,w)} = 1 \quad \forall\ (v,w)\in\ \varepsilon \\
(K8) & a_{(v,w)}\ =\ c_{(v,w)}\ -\ c_{(v,u)}\ +\ 4z_{(v,w)} \quad \forall\ (v,u),(v,w) \text{ subseqüentes em E'(v)} \\
& lb^{v}_{(v,w)},\ rb^{v}_{(v,w)}, lb^{w}_{(v,w)},\ rb^{w}_{(v,w)}\ \in\ \mathcal{N} \quad \forall\ (v,w)\ \in\ \varepsilon \\
& lb_{(v,w)},\ rb_{(v,w)}\ \in\ \mathcal{N} \quad \forall\ (v,w)\ \in\ \varepsilon \\
& a_{(v,w)}\ \in\ 0,...,4 \quad \forall\ (v,w)\ \in\ \varepsilon \\
& s^{d}_{(v,w)}\ \in\ 0,1 \quad \forall\ (v,w)\ \in\ E \\
& c_{(v,w)}\ \in\ 0,...,3 \quad \forall\ (v,w)\ \in\ E \\
& z_{(v,w)}\ \in\ 0,1 \quad \forall\ (v,w)\ \in\ E
\end{array}
\tag{3.11}
$$

Para montagem das restrições de posicionamento de arestas, foi preciso elaborar a relação enter a e c definidos nas seções anteriores. Para isso, foi preciso observar a relação destas variáveis em arestas subseqüentes no conjunto de adjacentes $\varepsilon(v)$.

$$
(K8)\quad a_{(v,w)}\ =\ c_{(v,w)}\ -\ c_{(v,u)}\ +\ 4z_{(v,w)} \quad \forall\ (v,u),(v,w) \text{ subsequentes em E'(v)} \tag{3.12}
$$

A variável z cria uma região factível suficiente, para que, na comparação de duas arestas subseqüentes, seja possível encontrar o valor de $a_{(v,w)}$ dados $c_{(v,w)}$ e $c_{(v,u)}$, sendo (v,w) e (v,u) as subseqüentes.

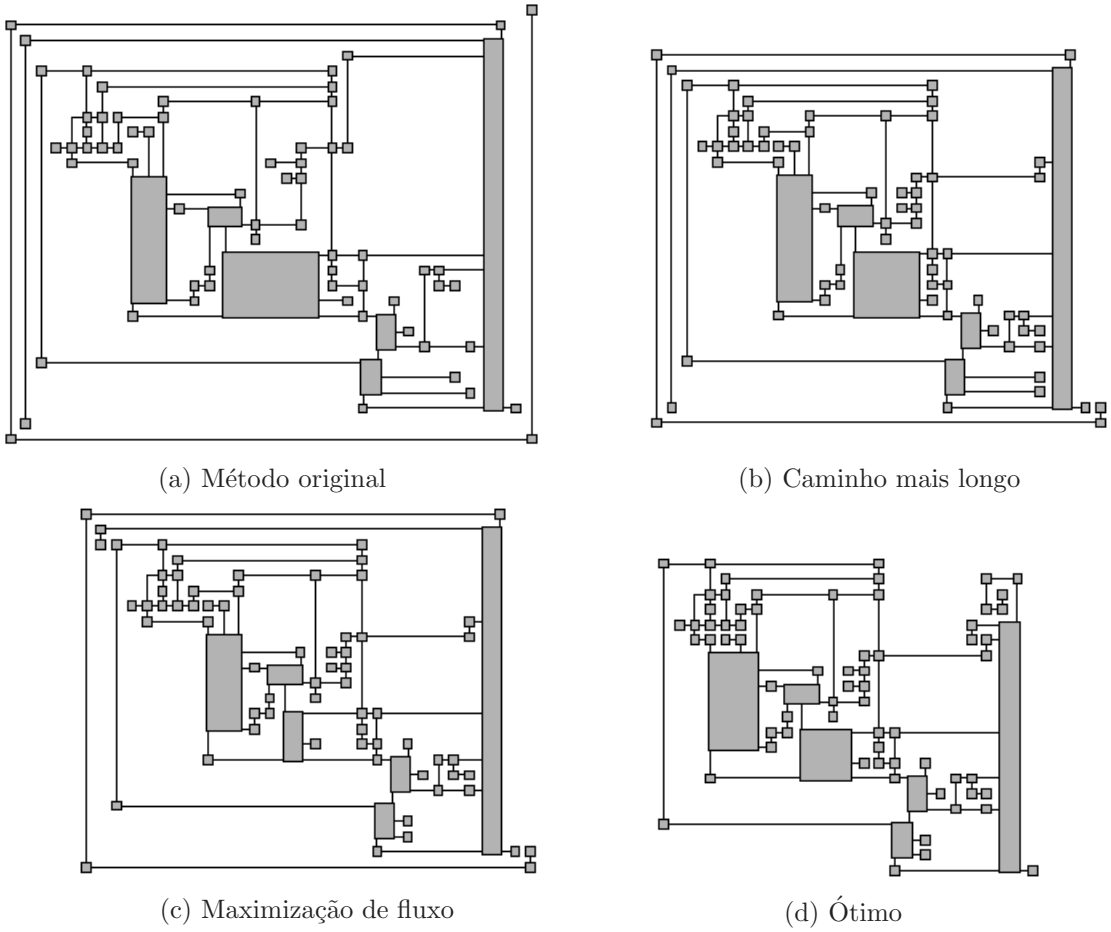
3.3 Compactação com Vértices de Tamanho Pré-Determinado

O objetivo da etapa de Compactação é minimizar a área do desenho ou a soma total dos comprimentos de arestas do grafo, enquanto se obedece a todas as informações de forma resultantes da etapa de Ortogonalização. Em desenho de grafos, somente são usadas heurísticas para compactar desenhos ortogonais em *grid*. Tamassia (1987) sugeriu refinar a forma de um desenho em um outro, composto somente por faces retangulares, introduzindo arestas artificiais. Se todas as faces são retangulares, o problema de compactação pode ser resolvido de forma aproximada em tempo polinomial utilizando-se algoritmos baseados em redes de fluxo de custo mínimo. Entretanto, em geral, esta solução está longe da ótima, considerando-se o grafo original, sem arestas artificiais. Outras heurísticas são baseadas na idéia de se fixar as coordenadas x ou y seguido de um passo de compactação unidimensional, em modo iterativo. Em uma compactação unidimensional, o objetivo é minimizar a largura, ou altura do desenho, estando preservadas as coordenadas da dimensão fixada. Esta abordagem também oferece um desempenho polinomial (Tamassia, 1987).

Klau and Mutzel (1999b) apresentam uma abordagem para a Compactação, baseada em sistema de minimização, que garante ser ótima. Lengauer (1990) e Patrignani (1999) provam que a maioria das versões de problemas em compactação em projetos de *layout* de grafos (VLSI) são *NP-hard*. Pesquisas em *layout* de circuitos VLSI concentram-se em métodos unidimensionais, pelo fato de que estudos anteriores a Klau and Mutzel (1999b), que abordam bidimensionalidade não são factíveis realmente. Lengauer (1990) estabelece o seguinte: "*A dificuldade da modelagem bidimensional está na determinação de como as duas dimensões devem interagir com a minimização da área*". Klau and Mutzel (1999b) apresentam uma abordagem que estabelece uma relação para o problema citado por Lengauer (1990). Tal abordagem provê a condição necessária e suficiente para se ter uma solução factível, dada uma instância de problema de compactação. A solução formulada por Klau and Mutzel (1999b) pode ser tratada via programação linear inteira.

A figura 3.7 mostra a diferença entre os principais métodos de Compactação de grafos. A figura 3.7a ilustra o método apresentado por Tamassia (1987). As figuras 3.7b e 3.7c mostram resultados de duas estratégias de compactação unidimensionais. Ambos métodos utilizam o método chamado *layout graphs*, sendo que o primeiro algoritmo é baseado em computação de caminho mais longo, e o segundo, em método de fluxo máximo. A figura 3.7d mostra o desenho compactado, baseado em PLI (Klau and Mutzel, 1999b).

O problema da compactação foi estudado por Klau and Mutzel (1999b) que definiram uma solução ótima. Eiglsperger and Kaufmann (2001) propuseram uma heurística para tornar a solução de complexidade linear no tempo, no pior caso. Tal método foi escolhido para implementação neste trabalho de mestrado, e também será descrito aqui neste capítulo, após a descrição do método ótimo, que serve como referência principal para o método rápido.



(a) Método original

(b) Caminho mais longo

(c) Maximização de fluxo

(d) Ótimo

Figura 3.7: Resultados de 4 métodos diferentes de Compactação

3.3.1 Compactação Ótima

A abordagem PLI da compactação é baseada em uma noção de descrição de forma para grafos (será descrita nesta seção). O que possibilita montar o sistema PLI é a tradução de informações gráficas em um modelo de otimização.

Descrição de Forma

Em um desenho ortogonal (em grid) Γ de um grafo G, os vértices são posicionados em pontos no grid (coordenadas inteiras) e as arestas são posicionadas nos caminhos (verticais e horizontais) dos pontos de grid. Chamamos de *simples*, um desenho ortogonal em grid que possui número de dobras em arestas e número de cruzamentos igual a zero. Todo desenho pode ser transformado em um desenho simples, criando-se vértices em dobras ou em cruzamentos, conforme mostrado na figura 3.8.

Definição 2 *O problema de compactação para desenhos ortogonais (CDO) é definido por: Dado um desenho ortogonal em grid simples Γ com uma representação ortogonal H, encontre um desenho Γ' com representação ortogonal H com soma total de arestas mínima.*

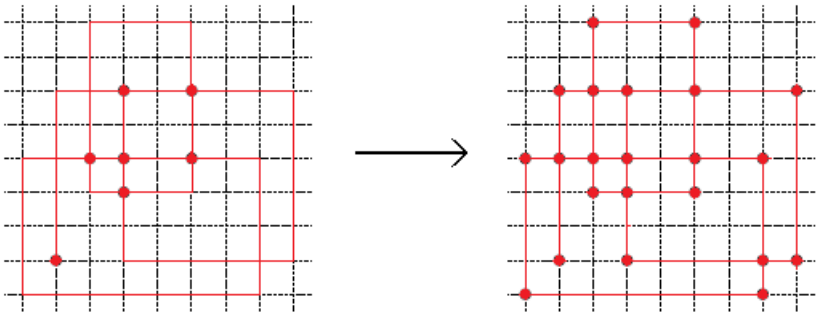


Figura 3.8: Transformação de um grafo para o modo simples.

O problema da compactação mostrou-se ser *NP-hard* (Patrignani, 1999), e por isso na prática, sempre se usou soluções unidimensionais que, no entanto, não traziam resultados satisfatórios.

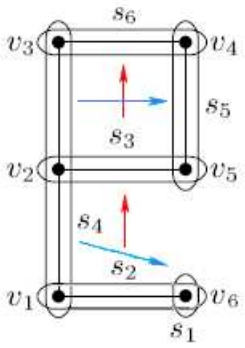
Seja Γ um desenho ortogonal em grid simples, de um grafo $G = (V, E)$. Isso induz a idéia de particionar o grafo em dois conjuntos de arestas, um horizontal E_h e outro vertical E_v. Um segmento é definido como sendo um grupo de arestas super-conectadas, ou grupos de arestas conectadas em linha, em qualquer um destes conjuntos vertical ou horizontal. São definidos, então, os conjuntos de segmentos para direção horizontal S_h, e um conjunto de segmentos para direção vertical S_v. Algumas observações para os segmentos:

1. Toda aresta é um sub-segmento, ou, $E_v \subseteq S_v$, $E_h \subseteq S_h$;
2. Cada vértice pertence a um segmento vertical e um horizontal, denotados $ver(v)$ e $hor(v)$, respectivamente;
3. Cada sub-segmento está contido em somente um segmento;
4. Os limites de um segmento s são dado da seguinte maneira: Sejam v_l, v_r, v_b e v_t os vértices, respectivamente, mais à esquerda, direita, abaixo e acima de s. Então, $l(s) = ver(v_l)$, $r(s) = ver(v_r)$, $b(s) = hor(v_b)$ e $t(s) = hor(v_t)$.

Uma descrição de forma, de uma representação ortogonal H é uma tupla $\omega = \langle D_h, D_v \rangle$ dos comumente chamados grafos de restrição. Ambos grafos são direcionados e definidos como sendo $D_h = (S_h, A_h)$ e $D_v = (S_v, A_v)$. Dessa forma, cada nó em D_h e D_v é um segmento, e as arestas são definidas como arcos, configurados desta maneira:

$$A_h = \{(l(e), r(e)) \mid e \in E_h\} \quad e \quad A_v = \{(b(e), t(e)) \mid e \in E_v\} \tag{3.13}$$

Cada segmento possui informações que definem, ou reforçam, a forma do desenho de um grafo. Um arco e em A_h que liga dois segmentos verticais $s_i = l(e)$ e $s_j = r(e)$, define que s_i será colocado mais à esquerda de s_j. Analogamente, se $s_i = b(e)$ e $s_j = t(e)$ definem um arco e em A_h, isso significa que s_i terá que ser colocado abaixo de s_j. E, em qualquer dos dígrafos da descrição da forma, um arco deverá ter ao final da compactação um tamanho mínimo de uma unidade de grid. A figura 3.9 exemplifica a caracterização da descrição da forma para um grafo.



i	$V(s_i)$	$E(s_i)$	$l(s_i)$	$r(s_i)$	$b(s_i)$	$t(s_i)$
1	$\{v_6\}$	$\{\}$	s_1	s_1	s_2	s_2
2	$\{v_1, v_6\}$	$\{(v_1, v_6)\}$	s_4	s_1	s_2	s_2
3	$\{v_2, v_5\}$	$\{(v_2, v_5)\}$	s_4	s_5	s_3	s_3
4	$\{v_1, v_2, v_3\}$	$\{(v_1, v_2), (v_2, v_3)\}$	s_4	s_4	s_2	s_6
5	$\{v_5, v_4\}$	$\{(v5, v_4)\}$	s_5	s_5	s_3	s_6
6	$\{v_3, v_4\}$	$\{(v_3, v_4)\}$	s_4	s_5	s_6	s_6

Figura 3.9: Caracterização de um grafo de restrições e a definição dos segmentos. Ambos $Ar = s_2, s_3, s_6$ e $Au = s_1, s_2, s_5$ estão destacados pelas setas vermelhas e azuis, respectivamente.

Completude

Para dois vértices v e w, é usada uma notação $v \xrightarrow{*} w$ se existe um caminho entre v e w. Descrições de forma possuem a seguinte propriedade: seja $\omega = \langle (S_h, A_h), (S_v, A_v) \rangle$ uma descrição de forma de um grafo. Para todo sub-segmento $s \in S_h \bigcup S_v$ os caminhos $l(s) \xrightarrow{*} r(s)$ e $b(s) \xrightarrow{*} t(s)$ estão contidos em $A_h \bigcup A_v$.

Definição 3 *Seja $(s_i, s_j) \in S \times S$ um par de segmentos. Podemos chamar tal par de* separado *se, e somente se, uma das condições a seguir se perdurar:*

1. $r(s_i) \xrightarrow{*} l(s_j)$
2. $r(s_j) \xrightarrow{*} l(s_i)$
3. $t(s_j) \xrightarrow{*} b(s_i)$
4. $t(s_i) \xrightarrow{*} b(s_j)$

Em desenhos ortogonais, somente uma destas condições é suficiente para caracterizar a separação.

Definição 4 *Uma extensão completa de uma descrição de forma $\omega = \langle (S_h, A_h), (S_v, A_v) \rangle$ é uma tupla $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ com as seguintes propriedades:*

1. $A_h \subseteq B_h, A_v \subseteq B_v$.
2. *B_h e B_v são acíclicos.*
3. *Todo par de segmento é* separado.

A definição 4 caracteriza o conjunto de soluções factíveis para o problema da compactação. Esta definição pode ser entendida com a idéia de que para ser considerada uma *extensão completa*, uma descrição de forma nunca deve possuir um segmento inalcançável

por qualquer um dos caminhos de arcos. Isso significaria uma indefinição do posicionamento deste segmento, no modelo de minimização da compactação. Klau and Mutzel (1999b) ainda mostram as condições básicas e as exceções para a definição da extensão completa da descrição da forma. Segundo eles, é sempre possível determinar uma extensão completa de uma descrição de forma, usando qualquer tipo de algoritmo de ordenação topológica (e.g. caminho mais longo ou algoritmos de fluxo máximo, no grafo dual à representação original). Posteriormente, Eiglsperger and Kaufmann (2001) definiram uma heurística que obtém a extensão completa em tempo linear.

3.3.2 Compactação Rápida

Eiglsperger and Kaufmann (2001) mostram em seu trabalho que é possível obter custo linear em tempo de execução para encontrar a completude em uma descrição de forma. As heurísticas usadas são baseadas em técnicas de decomposição retangular (Tamassia, 1987). O ponto inicial para a estratégia de decomposição retangular é observar se todas as faces de um desenho de grafo são retangulares. Para faces não retangulares, a idéia é subdividi-las em faces retangulares e resolver o problema localmente. A chave neste processo é identificar as faces não retangulares de maneira eficiente, e isso pode ser feito assumindo-se um padrão de ângulos nas faces. Para isso, ângulos de $90\,^\circ$ são denotados por '0' e ângulos de $270\,^\circ$ por '1'. Toda vez que se é encontrado um padrão '100' em uma face, esta é cortada neste ponto, e a varredura pelo padrão segue dentro da face, percorrendo todas as faces. A figura 3.10 ilustra esta idéia.

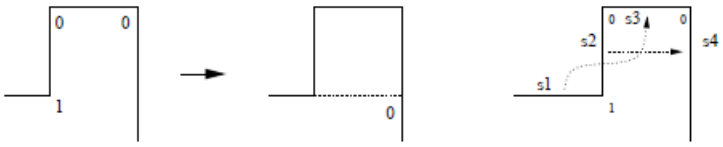


Figura 3.10: Decomposição de uma face em um retângulo e o resto da face.

Exatamente no ponto de definição do padrão '100' é que pode-se executar a heurística da completude. A figura 3.10 mostra o exemplo de seccionamento da face, que é seguido pela definição de arcos na descrição da forma ($s_1 \rightarrow s_3$ e $s_2 \rightarrow s_4$). A inclusão destes arcos pré-definidos a partir da varredura do padrão '100' possibilita a construção de uma descrição de forma completa.

Os algoritmos 1 e 2 mostram como montar a descrição completa.

Algoritmo 1: Define-Box - Função para gerar as arestas do grafo de restrições para os padrões da heurística.

Dados: Descrição de forma S, Direção d, Segmentos s_1, s_2, s_3, s_4
Resultado: A_h e A_v atualizados

```
1  S = ((S_h, A_v), (S_v, A_h));
2  se d = cima então
3  |  A_h ← A_h ⋃ (s_2, s_4),  A_v ← A_v ⋃ (s_3, s_1);
4  fim se
5  se d = baixo então
6  |  A_h ← A_h ⋃ (s_4, s_2),  A_v ← A_v ⋃ (s_1, s_3);
7  fim se
8  se d = esquerda então
9  |  A_h ← A_h ⋃ (s_1, s_3),  A_v ← A_v ⋃ (s_2, s_4);
10 fim se
11 se d = direita então
12 |  A_h ← A_h ⋃ (s_3, s_1),  A_v ← A_v ⋃ (s_4, s_2);
13 fim se
```

Algoritmo 2: Decompose - Varre as faces procurando os padrões da heurística.

Dados: Descrição de forma S, lista inicial de faces l
Resultado: Arestas em l

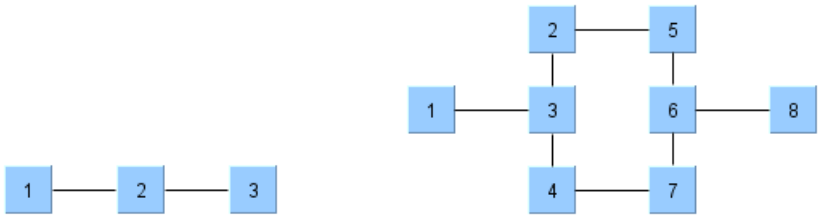
```
1 para todo Arestas em l faça
2 |  Procura por padrão, olhando arestas anteriores;
3 |  se padrão == 100 então
4 |  |  chame define-box para as arestas que formam padrão;
5 |  |  remova as arestas deste padrão da lista;
6 |  senão
7 |  |  mova a aresta para o fim da lista;
8 |  fim se
9 fim para todo
```

A execução destes algoritmos segue uma rotina precedente, que define os ângulos de curva, percorrendo as faces. Ao final da execução de *Decompose*, todos o padrões deverão ser processados, e o que resta é a minimização do grafo em um retângulo (perceba a retirada das arestas já processadas da lista inicial de arestas a percorrer). Desta forma garante-se que dois segmentos que deveriam ser conectados nos grafos restritos, vão se encontrar (na minimização do grafo).

Eiglsperger and Kaufmann (2001) mostram que este grupo de algoritmos calcula a descrição completa de forma de tamanho $O(n_e)$ (para n_e sendo o número de arestas do grafo) em tempo linear. Primeiramente, é preciso mostrar que a descrição completa tem tamanho $O(n_e)$. A descrição de forma inicial tem tamanho linear, de acordo com a fórmula de Euler.

Vértices com Tamanho Pré-Definido

Em seu trabalho, Eiglsperger and Kaufmann (2001) descrevem opções para forçar vértices com tamanho pré-definido. A opção adotada neste trabalho é a de *split* (explosão) dos vértices, ou seja: cada vértice que deve possuir altura e largura, é substituído por um conjunto de vértices e arestas formando um retângulo. O split, ou explosão, é realizado após a etapa de ortogonalização, pois é simples a criação dos ângulos que definem cada retângulo. Na implementação deste trabalho, rotinas automáticas varrem os vértices que correspondem a elementos especiais do circuito alimentador e substituem-os pelos retângulos específicos, pré-moldados (com ângulos de arestas já pré-definidos, segundo as especificações dos elementos especiais). As figuras 3.11, 3.12 e 3.13 mostram exemplos da técnica de explosão de vértices.



(a) Trecho de grafo simples.

(b) Mesmo trecho de grafo, com a explosão do vértice 2 do grafo simples.

Figura 3.11: Exemplo de caracterização do explosão de vértices. Além da inserção de vértices arestas formando um retângulo, é preciso criar também os ângulos de arestas, e as restrições necessárias para controlar as arestas que compõem os lados do retângulo. Desta forma é possível fazer o retângulo se comportar como um vértice de tamanho pré-definido.

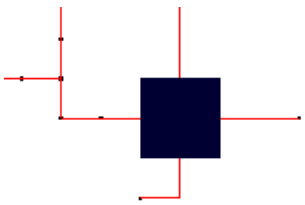


Figura 3.12: Trecho de desenho com exemplo de explosão de vértice que caracteriza uma chave seccionadora de 3 posições. O capítulo 5 mostra o que é preciso para caracterizar uma chave de 3 posições.

3.3.3 Sistema PLI da Compactação

O sistema linear inteiro que define a compactação é mostrado na equação 3.14, baseado em Eiglsperger and Kaufmann (2001):

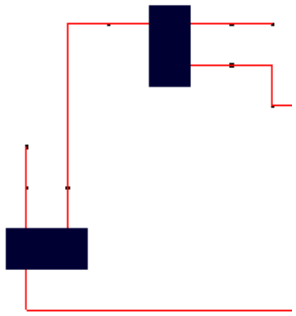


Figura 3.13: Trecho de desenho com exemplo de explosão de vértice que caracteriza uma chave seccionadora à gás. O capítulo 5 mostra o que é preciso para caracterizar uma chave à gás.

$$\begin{array}{rcccl} x_b - x_a & \geq & Length(e) & \forall e = (a,b) & \in A_h \\ y_b - y_a & \geq & Length(e) & \forall e = (a,b) & \in A_v \\ x_s & \geq & 0 & \forall s & \in S_h \\ y_s & \geq & 0 & \forall s & \in S_v \end{array} \tag{3.14}$$

Este sistema define que todos os segmentos têm que ter tamanho mínimo definido pela função *Length*. Esta função deve mapear as arestas que fazem parte de vértices explodidos e, assim, é possível definir um tamanho mínimo para os segmentos de tais vértices. Restrições análogas podem controlar o tamanho máximo dos segmentos e, assim, definir um tamanho exato para os vértices explodidos:

$$\begin{array}{rcl} x_b - x_a & \leq & LengthMax(e) \\ x_b - x_a & \geq & LengthMin(e) \end{array} \tag{3.15}$$

Como descrito anteriormente, a factibilidade deste sistema depende somente da completude da descrição de forma do grafo analisado (Eiglsperger and Kaufmann, 2001).

Capítulo 4

Desenho Hierárquico

Ao se tentar buscar um critério específico de desenho de grafos, que pudesse criar uma hierarquia entre os elementos (nodos), procurou-se estender a metodologia estudada para desenhos ortogonais e buscar uma que pudesse dar o enfoque necessário a este critério relevante. Segundo especificações do sistema de geração de sinóticos ortogonais da Light/RJ, o critério de hierarquia entre elementos dos circuitos alimentadores seria de extrema importância, já que estes objetivam oferecer informações baseadas em um fluxo de energia (gerador fonte para nodos receptores). Desta forma, foi decidido que os diagramas dos circuitos alimentadores seriam enquadrados numa classe mais específica de grafos, a dos grafos hierárquicos ou multi-níveis. Nesta classe, pode-se dividir o grafo em diversos subníveis, em que vértices de mesmo nível ocupam a mesma linha vertical (fluxo da esquerda pra direita, em direção horizontal e os níveis alinhados verticalmente) no desenho do grafo.

Um dos algoritmos mais utilizados no desenho de estruturas hierárquicas é o algoritmo de *Sugiyama* (Sugiyama et al., 1981). Este algoritmo apresenta na sua execução duas etapas bem definidas:

- Uma etapa de minimização de cruzamento entre arestas;

- Uma etapa em que se procura organizar a distância entre vértices de forma a garantir uma boa legibilidade do grafo.

Na primeira etapa, os cruzamentos são desfeitos alterando a ordem em que os elementos de um mesmo nível aparecem. Na segunda etapa, os vértices são reorganizados, buscando gerar um grafo mais compacto e com um bom grau de balanceamento entre os elementos conectados. Entretanto, mesmo antes destas etapas, pode-se considerar como uma pré etapa, o momento de se definir os níveis da hierarquia.

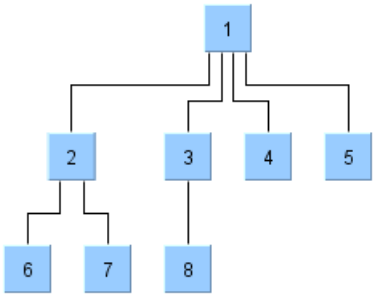


Figura 4.1: Exemplo de um diagrama hierárquico.

4.1 Definições Básicas

Uma hierarquia n-nível (Sugiyama et al., 1981; Sugiyama, 2002) é definida como um grafo direcionado (V, E), onde V representa o conjunto de vértices e E define o conjunto de arestas, que satisfazem as seguintes condições:

1. V é partido em n subconjuntos, tal que
$$V = V_1 \bigcup V_2 \bigcup ... \bigcup V_n \qquad \left(V_i \bigcap V_j = \emptyset,\ i \neq j\right) \tag{4.1}$$
onde V_i é o i-ésimo nível e n é o tamanho da hierarquia.

2. Toda aresta $e = (v_i, v_j) \in E$, onde $v_i \in V_i$ e $v_j \in V_j$, satisfaz $i < j$ e cada aresta em E é única.

3. E é partido em $n-1$ subconjuntos, tal que
$$E = E_1 \bigcup E_2 \bigcup ... \bigcup E_{n-1} \qquad \left(E_i \bigcap E_j = \emptyset,\ i \neq j\right) \tag{4.2}$$
onde $E_i \subset V_i \times V_{i+1},\ i = 1, ..., n-1$.

4. Uma ordem ς_i de V_i é dada para cada i, onde o termo "ordem" significa uma seqüencia de V_i, $\varsigma_i = v_1 v_2 ... v_{|V_i|}$ ($|V_i|$ denota o número de vértices em V_i).

A hierarquia n-nível é denotada $G = (V,\ E,\ n,\ \varsigma)$, onde $\varsigma = (\varsigma_1, ..., \varsigma_n)$. A hierarquia é dita *própria* se atender às condições 3 e 4. O desenho de uma hierarquia é chamado de *mapa*. Em um mapa, todos os vértices pertencentes ao i-ésimo nível V_i são posicionados de forma que ocupem a i-ésima linha de n linhas numeradas de cima para baixo. Para a abordagem de Sugiyama é necessário tratar em fase de pré-processamento o dígrafo de maneira a torná-lo acíclico.

4.2 Algoritmo Sugiyama

O algoritmo de Sugiyama está entre os principais algoritmos para desenho de grafos multi-níveis. O algoritmo trata somente grafos ditos próprios e pode ser dividido em três fases.

Distribuição em camada: Recebe um dígrafo G acíclico como entrada e os vértices de G são atribuídos nas camadas $L_1, L_2, ..., L_h$ tais que se (u, v) são os vértices extremos de uma aresta com $u \in L_i$ e $v \in L_j$ então $i < j$. No desenho final, cada vértice em camada L_i terá coordenada y igual a i. Em seguida são inseridos vértices fictícios ao longo das arestas que estão em mais de duas camadas, de maneira que o $span$ (que vale $i - j$) da aresta (u, v) com $u \in L_i$ e $v \in L_j$ não seja maior que 1. Assim, podemos definir que se todas as arestas que estiverem em mais de duas camadas possuírem vértices fictícios de maneira que $i = j + 1$, temos um "dígrafo próprio em camadas". A altura do dígrafo em camadas é o número de camadas (h) e a largura é o número de vértices na maior camada. O objetivo da atribuição em camadas é manter o dígrafo em camada o mais compacto possível (altura e largura a menor possível) e obter um dígrafo próprio em camadas com a inserção dos vértices fictícios, mas com critérios para que o número de vértices fictícios não seja grande. Este objetivo é necessário porque o tempo requerido nos passos seguintes a este é proporcional ao número total de vértices (falsos e reais). Dobras nas arestas do desenho final ocorrem somente nos vértices fictícios. O número de dobras afeta a legibilidade do desenho.

Redução de cruzamentos: Este passo recebe o produto da etapa de hierarquização como entrada e produz um novo dígrafo próprio em camadas no qual uma ordem é especificada para os vértices de cada camada. O número de cruzamentos entre arestas do desenho do dígrafo em camadas não depende da posição precisa dos vértices, mas sim da ordenação dos vértices dentro da camada. A ordem dos vértices nas camadas determina a topologia do desenho final e são escolhidas de maneira que o número de cruzamentos seja mantido o menor possível. Assim, o problema de redução de cruzamentos é um problema combinatorial relacionado com a escolha de uma ordenação apropriada dos vértices para cada camada, não da escolha geométrica de cada coordenada x para cada vértice. De fato o problema de minimização de cruzamentos entre arestas no dígrafo em camadas é NP-Completo, mesmo se ele contiver apenas duas camadas. Assim, uma variedade de heurísticas têm sido utilizadas para minimização de cruzamentos entre arestas em dígrafos em camadas.

Atribuição de coordenadas horizontais: Este passo recebe um dígrafo próprio em camadas, como entrada e produz coordenadas x finais para os vértices, preservando a ordenação e a redução de cruzamentos obtidos no passo anterior. Ao final deste passo, o desenho final é obtido representando cada aresta com um segmento de linha reta e então os vértices fictícios são removidos. Desta maneira, arestas longas podem ser representadas por linhas poligonais

A abordagem hierárquica também suporta alguns tipos de restrições. Por exemplo: dois dados vértices na mesma camada podem ser restritos por estarem próximos um do outro, no passo de redução de cruzamentos. Alguns critérios estéticos podem ser levados em conta durante o passo de atribuição da coordenada x, por exemplo: vértices podem ser alinhados para reduzir o número de dobras e podem, também, ser visualizados horizontalmente para enfatizar simetrias e podem ser empacotados, para reduzir a área.

A maioria dos problemas que ocorrem durante as fases desta abordagem são NP-hard e são conhecidos na literatura: *Feedback-arc Set* (Karp, 1972), *Precedence Constrained*

Multiprocessor Scheduling (Coffman Jr. and Graham, 1972), *2-layer Crossing Minimization* (Eades and Wormald, 1994), *Optimal Linear Arrangement* (Garey et al., 1974).

Heurísticas apropriadas têm sido desenvolvidas para a solução de todos os problemas citados e quase todos os softwares práticos para desenho de grafos utilizam esta abordagem. A maioria enriquecida por modificações necessárias para se atender a determinadas situações práticas como: vértices muito grandes, arestas em várias camadas, entre outras.

As seções a seguir detalham uma abordagem para o método Sugiyama definida por Eiglsperger et al. (2005).

4.2.1 Normalização e Atribuição de Níveis

Em um dígrafo acíclico $G = (V, E)$, é definido que $L_1, .., L_h$ são as partições de V com $L_i \subset V$, $1 \leq i \leq h$ e $\bigcup_{i=1}^{h} L_i = V$ (h é o número de níveis). Tal partição é chamada de *layering* de G, se para todas as arestas $e = (v, w)$ com $v \in L_i$ e $w \in L_j$ então $i > j$. O *span* de uma aresta e é $i - j$. O número de vértices em uma camada L_i é denotado por n_i. Em um desenho em camadas, todos os vértices $v \in L_i$ são desenhados em uma linha horizontal (mesma coordenada y). Assim, o passo de atribuição em camadas atribui a cada vértice v uma coordenada y. O *layering* é chamado dígrafo próprio em camadas se o $span(e) = 1$ para todas as arestas $e \in E$. Na maioria das aplicações, as camadas dos vértices podem ser atribuídas arbitrariamente e, em alguns casos, esta atribuição em camadas eventualmente pode ser parte da entrada.

Para arestas $e = (u, v)$ com $span(e) > 1$ (o que significa que a aresta e atravessa mais que duas camadas) cujos extremos u e v estão nas camadas L_i e L_j substituí-se a aresta e por uma cadeia de vértices fictícios $u = d_i, d_i + 1, ..., d_{j-1}, d_j = v$ em que quaisquer dois vértices fictícios consecutivos são conectados por uma aresta falsa. Vértices d_k para $i \leq k \leq j$ são colocados na camada L_k. Este processo é chamado normalização e o resultado é um dígrafo normalizado $G_N = (V_N, E_N)$. Com esta construção, a próxima fase inicializa-se com um dígrafo próprio em camadas.

A implementação de Gansner et al. (1993) propõe a minimização da soma dos tamanhos das arestas, e desta forma, o número de vértices falsos também é minimizado (Frick, 1997). O algoritmo para minimização do número de vértices fictícios é um método simplex em rede e nenhum limite de tempo polinomial foi provado para ele, mas algumas heurísticas de tempo linear para este problema funcionam bem na prática (Healy and Nikolov, 2002; Sander, 1999). No pior caso $|V_N| = O(|V||E|)$ e $|E_N| = O(|V||E|)$.

A finalização deste processo se dá ao transformar os vértices falsos em pontos de dobras de arestas.

4.2.2 Redução de Cruzamentos

Nesta etapa, utiliza-se listas ordenadas como estruturas de dados para armazenamento dos elementos do grafo. Os vértices dentro de cada camada L_i são armazenados na lista ordenada, da esquerda para direita, na correspondente linha horizontal. Esta ordenação é chamada ordenação em camada. A camada será sempre identificada pela lista corres-

pondente L_i. A ordenação dos vértices nas camadas adjacentes L_{i-1} e L_i determinam o número de cruzamentos entre arestas, com seus extremos em ambas as camadas.

A redução de cruzamentos é usualmente feita por uma varredura, camada a camada, de forma que cada passo minimize o número de cruzamentos entre arestas, para um par de camadas adjacentes. Esta varredura é feita da seguinte forma: começa-se escolhendo uma ordem arbitrária para os vértices para a primeira camada L_1. Assim, interativamente, enquanto a ordenação dos vértices da camada L_{i-1} é mantida fixa, os vértices de L_i são colocados em uma ordem que minimiza os cruzamentos. Este passo é chamado *minimização de cruzamentos em duas camadas* e é repetido para $i = 2, ..., h$. Após se ter processado a camada mais inferior, inverte-se a direção da varredura e segue-se de baixo para cima. Estes passos são repetidos até que mais nenhum cruzamento possa ser eliminado para um certo número de iterações.

Para o problema específico de minimização de cruzamentos entre duas camadas, são geralmente utilizadas heurísticas (Battista et al., 1999):

- *Método da Ordenação*: Uma maneira de se resolver o problema da minimização de cruzamentos em duas camadas é ordenar os vértices na camada L_2 em uma ordem que minimiza o número de cruzamentos. Isto pode ser feito através de algoritmos simples, que requerem cálculo do número de cruzamentos. Segundo Battista et al. (1999), com algoritmos simples é possível obter tempo de execução na casa de $O(|E|^2)$.

- *Método do Baricentro* (Sugiyama et al., 1981): Este método é baseado no princípio do baricentro, onde a medida de um vértice v é o baricentro (média) das posições de seus vizinhos na camada acima. Obtém-se as posições dos vértices em L_i pela ordenação dos mesmos de acordo com suas medidas.

- *Método da Mediana* (Eades and Wormald, 1994): Possui o princípio básico de se trocar posições de vértices de um nível se esta troca minimizar o número de cruzamentos. Possui complexidade quadrática, pior em relação à linear obtida nos métodos descritos anteriormente. Ela é principalmente utilizada como otimização local das heurísticas mediana e baricentro na iteração da varredura na camada

Existem alguns poucos resultados documentados baseados na qualidade das heurísticas apresentadas anteriormente. Um deles diz que a heurística do baricentro, bem como da mediana dá a solução sem cruzamentos se a mesma existir (Battista et al., 1999).

O método de medição de qualidade de um passo de redução de cruzamentos é a contagem destes cruzamentos. Isto é dado como um sub-problema desta etapa, e é comumente tratado por um algoritmo de varredura ingênuo que gasta tempo $O(|E'| + |C'|)$, onde $|E'|$ é o número de arestas entre as duas camadas e $|C'|$, o número de cruzamentos entre estas arestas (Sander, 1999). Este algoritmo sofreu melhorias recentemente para $O(|E'|log|V'|)$ por Waddle and Malhortra (1999). Barth et al. (2002) mostram uma descrição bem simples do algoritmo e com a mesma complexidade de tempo: sejam L_i e L_{i+1} duas camadas adjacentes com ordenação $v_1, ..., v_p$ e $w_1, ..., w_q$ respectivamente. As arestas entre ambas as camadas são ordenadas lexicograficamente de maneira que $(v_i, w_j) < (v_k, w_l)$ se e

somente se $i < k$ ou $i = k$ e $j < l$. Seja $e1, ..., er$ uma seqüência de arestas ordenadas lexicograficamente e j_m o índice do vértice destino de e_m. Uma inversão na seqüência $j_1, ..., j_r$ é um par j_k, j_l com $k < l$ e $j_k > j_l$. Cada inversão corresponde a um cruzamento de arestas entre ambas as camadas, portanto deve ser contabilizado. O número de inversões é calculado por meio de uma estrutura de dados eficiente, chamada *árvore acumuladora T*. A estrutura de dados pode ser facilmente estendida para suportar contagem de cruzamentos com arestas pesadas, as quais serão relevantes posteriormente.

4.2.3 Atribuição de Coordenadas Horizontais

Esta etapa se incube de definir as coordenadas x para cada vértice, uma vez que todos eles já possuem os níveis definidos (coordenadas y) e uma ordenação. Existem dois objetivos a serem considerados para que se obtenha bons desenhos. No primeiro, os desenhos devem ser compactos e no segundo, as arestas devem ser "tão verticais quanto possível". Falhas no segundo objetivo podem produzir muitas dobras desnecessárias, o que reduz a legibilidade do desenho.

O problema de atribuição de coordenadas horizontais pode ser modelado por um sistema linear:

$$\min \quad \sum_{(v,w)\in E} \Omega(v,w).|x(v) - x(w)| \tag{4.3}$$

Sujeito a:

$$x(b) - x(a) \ \geq \ \delta(a,b) \quad a,b \text{ vértices consecutivos em } L_i, \ 1 \ \leq \ i \ \leq \ h \tag{4.4}$$

$\Omega(v,w)$ denota a prioridade para se desenhar a aresta (v,w) vertical e $\delta(a,b)$ denota a menor distância entre os vértices consecutivos a e b. O programa linear pode ser interpretado como um problema de grau de atribuição na compactação de um grafo $G_a = (V_n, (a,b)$: a,b consecutivos em $L_i, 1 \ \leq \ i \ \leq \ h)$ em função do tamanho δ. Cada grau de atribuição válido corresponde a um desenho válido. A função objetivo, mostrada acima, pode ser modelada pela adição de vértices e arestas ao grafo G_a (Gansner et al., 1993).

Uma outra abordagem seria o uso de modelo de segmentos lineares, em que cada aresta é desenhada como polilinhas (de acordo com o valor do $span(e)$) com pelo menos três segmentos (para $span(e) > 2$). Neste caso o segmento do meio é sempre desenhado na vertical. Em geral, desenhos de segmentos lineares têm menos dobras, mas precisam de mais área do que desenhos em outros modelos. Existe um número razoável de algoritmos propostos para este modelo (Sander, 1999; Brandes and Köpf, 2002).

A abordagem de Brandes and Köpf (2002) é uma heurística baseada no caminho mais longo para atribuição de coordenada horizontal em grafos direcionados e acíclicos. Ela produz segmentos verticais internos e bons desenhos balanceados em tempo linear. Primeiro se tenta alinhar um vértice tanto com seu vizinho médio mais acima quanto com o mais abaixo. Vértices alinhados compartilham o mesmo vértice no grafo compactado e assim

se obtém a mesma coordenada x. Existem três tipos de conflitos de alinhamento, conflito tipo 0, que aparece entre dois segmentos não-internos (um segmento não-interno tem pelo menos um vértice não-fictício em seu extremo), conflito tipo 1 que aparece entre um segmento não-interno e um segmento interno e conflito do tipo 2 que aparece entre dois segmentos internos. Conflitos do tipo 1 são sempre resolvidos em favor do segmento interno. Indiferentemente se os vértices estão alinhados com seus vizinhos medianos superior e inferior, conflitos de alinhamento podem ser resolvidos utilizando os conceitos de moda tanto mais à esquerda quanto mais à direita. Assim, existem quatro possíveis combinações para alinhamento de vértices. Para cada combinação as coordenadas horizontais são calculadas utilizando-se o algoritmo do caminho mais longo. Finalmente, as quatro atribuições de coordenadas resultantes são combinadas para se obter um desenho balanceado.

4.2.4 Desempenho do Algoritmo de Sugiyama

Uma característica importante da abordagem Sugiyama é que a sua complexidade é altamente dependente do número de vértices fictícios inseridos. Embora este número possa ser minimizado eficientemente, ela pode ainda ser da ordem de $O(|V||E|)$ (Frick, 1997). Assumindo-se que as etapas são desempenhadas com as melhores abordagens, pode-se definir que o método como um todo tem complexidade de tempo $O(|V||E|log|E|)$ e usa espaço $O(|V||E|)$.

Capítulo 5

Biblioteca de Geração de Esquemáticos Ortogonais

Este capítulo mostra os detalhes de projeto e implementação dos sistemas de geração automática de sinóticos desenvolvidos para a Light/RJ. Resultados obtidos são mostrados no capítulo 6.

5.1 Sistema de Geração de Diagramas Ortogonais

A necessidade de modernização das concessionárias de energia, aliada à questão da qualidade de fornecimento de energia elétrica, que atualmente é controlada pela Agência Nacional de Energia Elétrica - ANEEL, tem levado as empresas do setor, principalmente as de distribuição de energia elétrica, a buscar novos mecanismos para melhorar a qualidade dos serviços prestados aos seus consumidores.

A utilização de métodos computacionais avançados permitiu uma maior otimização dos serviços, propiciando o crescimento do interesse pelas concessionárias por novas tecnologias, principalmente quanto aos recursos oriundos dos Sistemas de Informação Geográficas (SIG).

É importante integrar a estes sistemas ferramentas que permitam à área de Operação recursos para visualização e manipulação eficiente dos elementos que caracterizam a rede de distribuição: trechos de rede, chaves de seccionamento, disjuntores, etc.

O projeto de P&D vinculado a este trabalho de mestrado consiste na geração automática de diagramas ortogonais compactos, baseados no Sistema de Informações Geográficas (SIG), permitindo visualização de circuitos de forma simplificada pelos operadores e a possibilidade de inclusão de novas funcionalidades. Também facilitará as atividades da área de Planejamento.

Atualmente a construção desse tipo de representação é viabilizada apenas através de intervenção humana. Como a quantidade de representações é usualmente extensa, e estas sofrem alterações constantes, a manutenção deste conjunto de dados através de ação humana contribui para a geração de inconsistências e defasagem nas representações, com

impacto negativo sobre as ações de manutenção e restabelecimento do sistema.

Assim, é importante criar os esquemáticos (diagramas ortogonais compactos) de forma automática para operação/planejamento, a partir das informações obtidas do SIG. As informações da topologia da rede já são extraídas na atualidade do SGBD do SIG por ferramenta existente na Light, mas a geração automática dos esquemáticos esbarra nos problemas de representação gráfica de uma forma amigável ao operador.

Requisitos, critérios e as soluções para o desenvolvimento da ferramenta de desenho automático de ortogonais são descritas a seguir.

5.1.1 Requisitos Para o Desenho de Circuitos de Redes de Transmissão

Nessa seção analisaremos as características específicas dos diagramas dos circuitos associados a sistemas de distribuição. A figura 5.1 mostra o diagrama ortogonal de um circuito de rede de distribuição gerado pelo sistema SGD da Light. Pela estrutura do desenho gerado, podemos inferir que o algoritmo utilizado para a geração deste desenho é baseado na estratégia dividir para conquistar (Reingold and Tilford, 1981; Walker, 1990). De fato, o grafo do circuito, apesar de não ser uma árvore (existem dois ciclos no diagrama) está representado como um desenho *hv* de árvore binária.

Os problemas desta representação são evidentes:

- A sua baixa compactação: como o diagrama está pouco compactado, o operador necessita navegar por várias telas para ter acesso aos elementos do circuito (o diagrama não "cabe" em uma única tela com a resolução necessária.
- A existência dos ciclos no grafo faz com que o desenho por uma árvore *hv* não seja adequado pois, apesar do grafo ser planar, existem cruzamentos de arestas gerados pelo desenho dos ciclos.

A figura 5.2 apresenta outro desenho para o mesmo circuito da figura 5.1, agora gerado por um desenhista, usando um programa de Projeto Assistido por Computador.

Neste diagrama, as seguintes características são visíveis:

- A representação é muito mais compacta: numa única tela tem-se resolução suficiente para visualizar todo o circuito. Como o desenhista não tem nenhuma das limitações associadas ao desenho *hv*, pode distribuir os elementos do circuito de forma mais apropriada.
- Os laços continuam existindo, mas foram eliminados os cruzamentos desnecessários. O desenhista distribuiu os elementos de modo que os laços ficassem localizados em pequenas regiões do circuito.

O desenho é muito mais fácil de compreender e sua utilização em um quadro sinótico leva o sistema a ter uma usabilidade muito melhor que utilizando o diagrama anterior. É

Figura 5.1: Diagrama ortogonal da LSA - Cavado da Light, gerado pelo sistema SGD.

importante enfatizar algumas das características dos desenhos gerados pelos desenhistas e que são considerados adequados ao domínio de aplicação.

1. As convenções ortogonal e planar foram utilizadas. A convenção grid pode ser utilizada em um sistema de geração automática de diagramas ortogonais.

2. Utilizar os seguintes critérios estéticos

   - minimizar cruzamentos de arestas;
   - minimizar área e utilizar razão de aspecto próxima à da tela do operador;
   - minimizar comprimento das arestas;
   - minimizar número de dobras

3. Quanto às restrições, as seguintes são comuns em tipos de desenho tais como os mostrados:

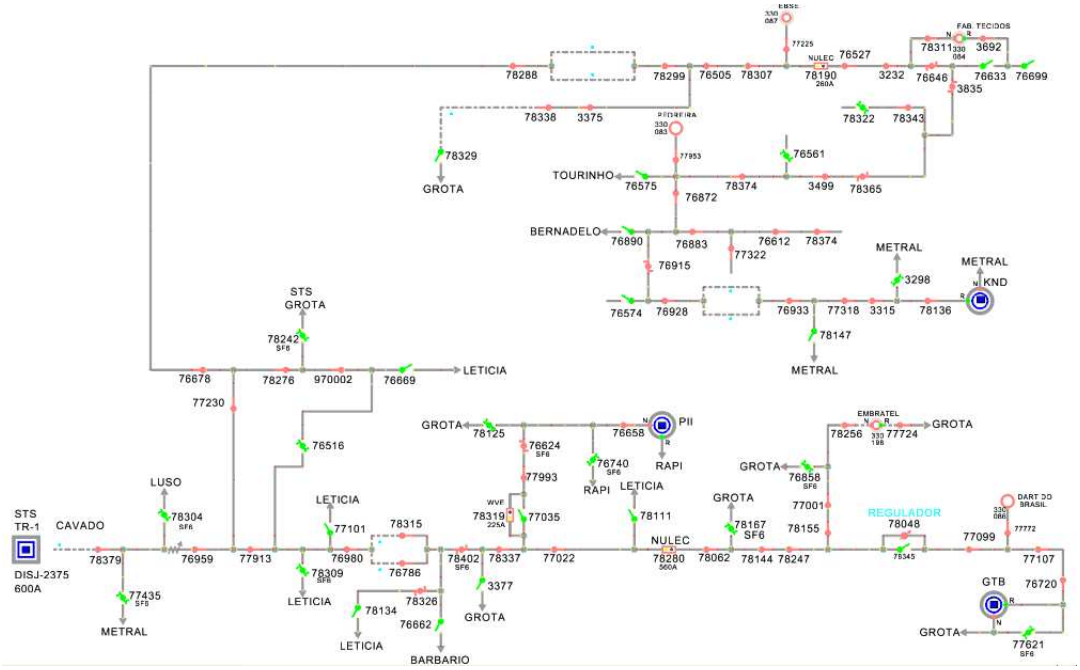


Figura 5.2: Diagrama ortogonal da LSA-Cavado da Light, gerado por desenhista.

- Alguns elementos são localizados em posições específicas. Por exemplo, a alimentação dos circuitos (saída das subestações) está sempre localizada à esquerda do desenho;
- Alguns elementos são agrupados;

Na seção a seguir, será detalhada a ferramenta implementada para gerar os sinóticos dos circuitos alimentadores da Light. Essencialmente, serão mostrados os requisitos do sistema, as soluções implementadas para integração da ferramenta ao sistema de mapas da Light, e também, as soluções construídas para resolução do problema de desenho de grafos.

5.1.2 Componentes da Solução Computacional

A ferramenta para desenho ortogonal de circuitos alimentadores da Light foi construída com o uso da metodologia topologia-forma-métrica e da metodologia hierárquica (descritas em capítulos anteriores). Dessa forma, a ferramenta pode gerar resultados com propriedades distintas que puderam ser selecionados no intuito de se definir uma metodologia que melhor se adequa ao tipo de circuito desenhado.

A arquitetura da plataforma desenvolvida pode ser descrita pelos seguintes módulos organizados na figura 5.3:

- *Módulo Controlador Geral*: neste módulo são manipulados todos os recursos da plataforma;
- *Módulo de Leitura*: aqui estão as rotinas para leitura das informações de grafos do banco de dados e rotinas para o pré-processamento das informações lidas.

- *Módulo de Algoritmos de Desenho*: neste módulo são encapsuladas todas as rotinas de desenho de grafos, divididos nas grandes áreas de desenho ortogonal e desenho hierárquico.

- *Módulo de Algoritmos de Saída*: compreende as rotinas que escrevem no formato de saída definido, os dados de desenho dos grafos roteados dos circuitos.

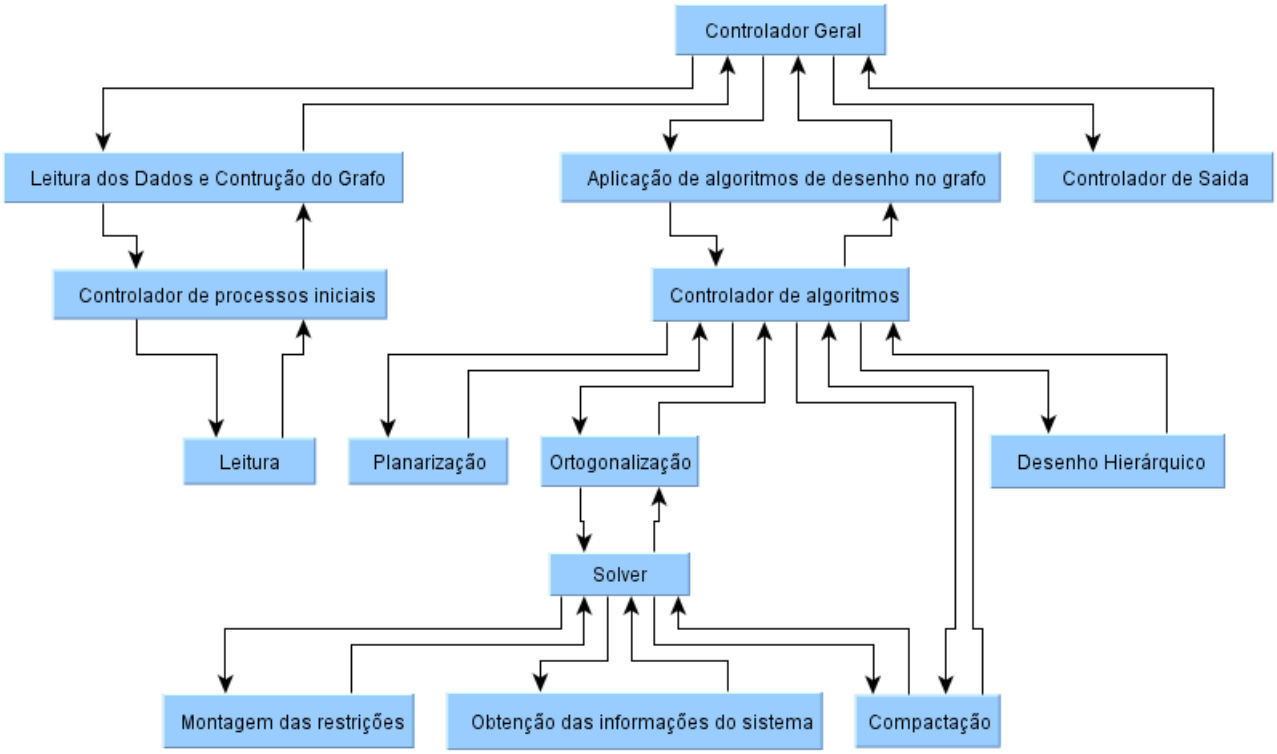


Figura 5.3: Arquitetura da plataforma implementada para leitura dos dados dos circuitos alimentadores e geração dos sinóticos.

Leitura dos Dados da Rede

O processo de construção de um grafo é dividido em duas etapas: leitura do banco de dados com as informações de conexão entre os diversos elementos que compõem o diagrama unifilar associado a um determinado alimentador e montagem do grafo a partir destas informações. A leitura é feita buscando-se, através de consultas SQL, os dados contidos nas tabelas do banco de Dados. Tais dados preenchem as estruturas correspondentes a cada tipo de elemento na plataforma e o grafo então pode ser montado, a partir das informações de conectividade entre os elementos.

Em alto nível, o algoritmo 3 mostra o fluxo de atividades para leitura dos dados.

Algoritmo 3: Controlador de Grafos.

Dados: Acesso ao Banco de Dados com informações sobre os componentes da rede
Resultado: Estrutura de grafo montada, a partir dos dados de ligação entre os elementos

1 Identifique os alimentadores a serem desenhados;
2 Faça a leitura dos alimentadores;
3 Armazene os elementos a serem desenhados em estruturas especiais;
4 Atribua identificadores aos elementos da linha;
5 Monte uma lista de vértices a partir da informação armazenada;
6 Monte uma lista de arestas a partir da informação armazenada;
7 Construa o grafo a partir das listas de vértices e arestas;

No algoritmo 3 são identificados os elementos especiais da linha, os quais serão desenhados de forma especial, ou utilizando algum processamento único, ou grupo de restrições únicas na etapa de ortogonalização. Os elementos dos circuitos alimentadores da Light são:

- Alimentador: é um elemento especial representado por um vértice que será a raiz, ou, o início do grafo - os diagramas dos circuitos alimentadores são representados por dígrafos que permitem somente alguns loops, como exceção. Tal característica deve explicitar o fluxo de energia sendo transmitida, pelo circuito.

- Cabos: elementos que unem dois vértices (quaisquer tipos), ou dois cabos, ou mesmo um vértice a um cabo. Possuem noção de direção (fonte - nó). São representados como as arestas do grafo.

- Vértices de ligação: são vértices criados para determinar uma ligação entre cabos (nó de um cabo anterior, ligado à fonte de um cabo posterior).

- Chaves Seccionadoras Simples: são representadas somente por um vértice simples, entretanto, com tamanho (largura e altura) definidos diferentes de 1 unidade.

- Chaves Seccionadoras de 3 Posições: grupos de chaves simples, que formam padrões de ligação definidos e catalogados. Na ferramenta, este tipo de grupo é sempre identificado e representado por 1 vértice somente, com dimensões especiais. A figura 5.4 mostra as configurações possíveis para uma chave de 3 posições.

- Chaves à Gás: grupos de chaves simples que também formam um padrão definido. São representados por 1 vértice de tamanho e forma especiais. A figura 5.5 mostra a especificação deste tipo de chave.

De acordo com especificações fornecidas (Indra, 2007), todo elemento possui uma tabela única no banco de dados. Dentre as informações de descrição de um elemento, estão as informações de coordenadas físicas, extraídas do sistema georeferenciado. Através do mapeamento das coordenadas de cada elemento é possível varrer o banco de dados e obter o grafo (ligação dos vértices e arestas).

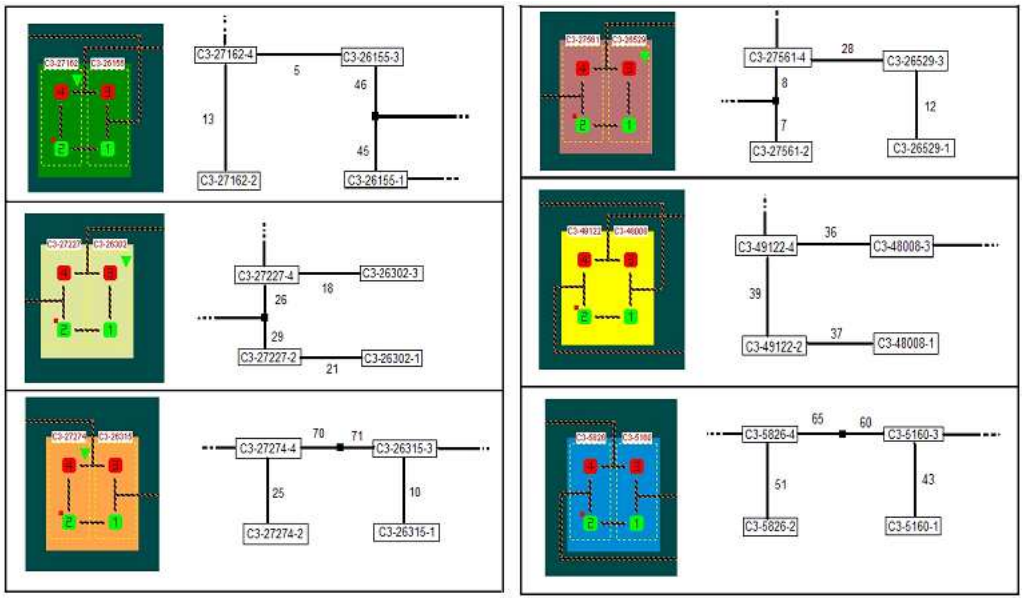


Figura 5.4: Exemplos de configurações previstas para chaves de 3 posições.

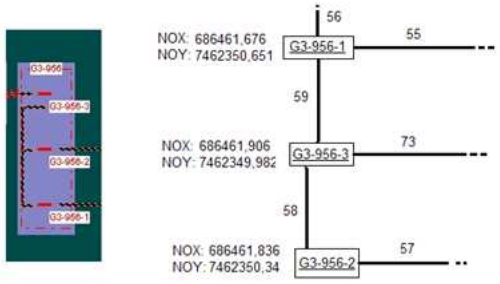


Figura 5.5: Exemplo de configuração prevista para chaves à gás. Este tipo de chave pode conter 3 ou 4 nodos internos (conseqüentemente, 3 ou 4 arestas adjacentes).

Mais detalhes sobre os procedimentos técnicos e as especificações do banco de dados e das estruturas usadas na plataforma para gerenciar a leitura dos elementos do circuito, estão presentes no relatório técnico (Mesquita and Terra, 2007).

Implementação dos Métodos Automáticos

A plataforma de geração de sinóticos foi desenvolvida para executar dois tipos de desenho de grafos. Uma das abordagens seguidas e implementadas é a baseada em Eiglsperger et al. (2000) e Klau and Mutzel (1999b) que foi descrita no capítulo 3. Esta abordagem visa a geração de desenhos minimizando-se o número de cruzamentos entre arestas e a área do desenho (respectivamente, nas etapas de ortogonalização e compactação). A segunda abordagem é a hierárquica, implementada segundo Sugiyama (2002). Esta abordagem possui a característica de oferecer desenhos com intenção de fluxo mais visíveis. Desenhos seguindo a abordagem ortogonal clássica podem mitigar a percepção, ou a idéia, de sentido de fluxo do grafo. Desenhos com abordagem hierárquica não possuem área minimizada.

Metodologia Ortogonal Clássica

Para implementar a abordagem topologia-forma-métrica, foram utilizadas algumas bibliotecas. O problema de planarização foi resolvido segundo a abordagem incremental de inserção de arestas (Battista et al., 1999). Para isso, foi utilizada a GTAD (*Graph Toolkit for Algorithms and Drawings*), desenvolvida em uma dissertação de mestrado na UFMG por Melo (2007). Foram utilizadas as estruturas internas da GTAD que representam um grafo para utilização da etapa modular da planarização. Nesse momento, a estrutura de grafo da plataforma é traduzida em termos do grafo da GTAD, e como resultado, é gerada uma estrutura auxiliar, que define a ordem das arestas adjacentes, para cada vértice. Tal estrutura auxiliar é especificada como um Hash de listas de ponteiros (ou identificadores). Cada posição do Hash significa uma entrada de vértice do grafo. A lista atrelada dá a ordenação das arestas adjacentes a este vértice. Com isso é possível obter a informação de planarização do grafo, e aproveitá-la como entrada da etapa de ortogonalização.

Na plataforma de geração de sinóticos, a ortogonalização foi desenvolvida segundo algumas restrições definidas como requisitos dos desenhos da Light. Posteriormente serão citadas as restrições que compõem a ortogonalização. Para trabalhar com os sistemas lineares montados segundo os capítulos anteriores, foi utilizada uma biblioteca de programação linear, cujo *solver* é comumente chamado de *lpsolve* (Team, 2003). O *lpsolve* possui manutenção e supervisão dirigidas por Kjell Eikland e Peter Notebaert sendo constantemente atualizado (está atualmente sob versão 5.5.0.14). Ferramentas como este solver recebem como entrada uma listagem das restrições do problema (em sintaxe própria) bem como uma matriz de coeficientes para modelagem das equações e inequações do problema. Isso significa que a inserção ou retirada de restrições é condicionada somente por inserção ou retirada de linhas na matriz de coeficientes, e da descrição formal da restrição na sintaxe do solver. A escolha deste solver foi devido a:

- Ser baseado em software livre, de forma a evitar custos oriundos de sua aquisição;
- Eficiência, tendo em vista a complexidade dos problemas de PLI.
- Disponibilidade sob forma de biblioteca, disponibilizando uma API (*Application Programming Interface*) de programação em C++.

A montagem das restrições do sistema no solver é feita de maneira análoga para todas. A diferença básica é o sujeito alvo de varredura: vértices, arestas ou as faces do grafo. O algoritmo 4 mostra um exemplo de preenchimento das estruturas do sistema lpsolve, segundo uma restrição advinda do método Kandinsky, dada equação 5.1.

$$\sum_{(v,w)\in\ \varepsilon(v)} a_{(v,w)} = 4 \quad \forall\ v\ \in\ V \tag{5.1}$$

Algoritmo 4: Montagem da Primeira Restrição.

Data: Matriz de restrições R com função objetivo adicionada, estrutura auxiliar para armazenamento de uma linha $linevet$

Result: Matriz de restrições R com função objetivo e 1ª restrição adicionadas

```
1  forall the v(i) do
2      forall the f(j) do
3          forall the e(k) ∈ f(j) do
4              if NoFonte(e(k)) == v(i) then
5                  Adicione os coeficientes de restrição à linevet;
6              endif
7          endfall
8      endfall
9      Adicione 4 à linevet;
10     Adicione linevet à primeira linha vazia de R;
11 endfall
```

Esta restrição indica que, para cada vértice, a soma do valor de a de cada aresta adjacente sempre tem que ser igual a 4. Para executar esta condição, são percorridos todos os vértices, e em cada vértice, são recuperadas as arestas adjacentes. Para estas arestas, são colocados os coeficientes relacionados a restrição. A estrutura $linevet$ é usada para mapear todas as posições da matriz de coeficientes, recuperando-se as posições de cada vértice ou aresta, em cada grupo de linhas que compreende a restrição.

De maneira análoga, as outras restrições são colocadas ao final da matriz, concatenando-se as linhas para que o sistema completo possa ser enviado ao solver.

Metodologia Hierárquica

A implementação da metodologia hierárquica foi feita utilizando-se uma biblioteca própria para execução do algoritmo de Sugiyama (Sugiyama, 2002). A OGDF (*Open Graph Drawing Framework*) (Chimani et al., 2007) é uma biblioteca desenvolvida em C++ que implementa o método e é licenciada sob os termos da GNU General User Licence versões 2 e 3. A figura 5.6 mostra o diagrama UML das classes envolvidas com a implementação do Sugiyama na OGDF. Tais estruturas foram utilizadas para tradução do grafo lido a partir do banco de dados, e a obtenção das coordenadas dos elementos (cabos - arestas - e demais elementos - vértices).

Algumas estruturas devem ser detalhadas:

- **Classe Graph**: É a classe que representa o grafo em si. Encapsula as informações sobre os nós e arestas do grafo que representa.
- **Classe GraphAttributes**: É a classe de acesso aos atributos do grafo. É através dela que utilizamos a estrutura do grafo. Ela atua como uma interface entre o usuário e o grafo relacionado a ela, permitindo gerenciar os atributos dos nós e das arestas, por exemplo.

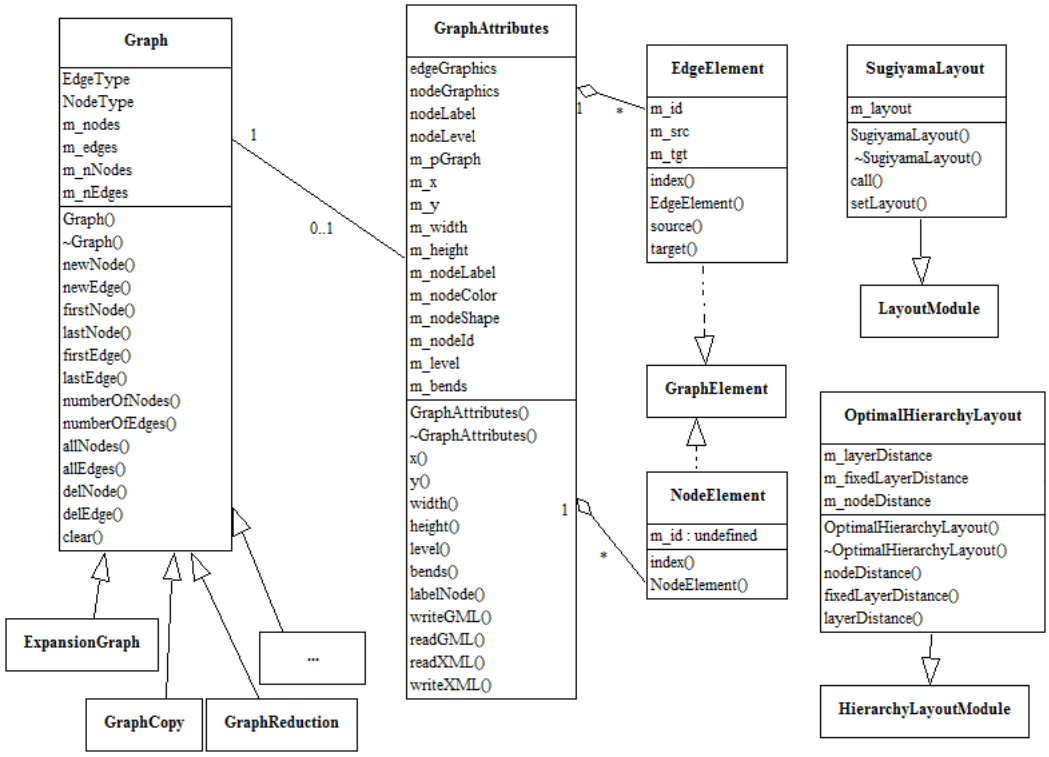


Figura 5.6: Diagrama de Classes da biblioteca OGDF. O diagrama só apresenta as classes, funções membro e atributos utilizados neste projeto.

- **Classe EdgeElement**: É a classe que representa uma aresta.
- **Classe NodeElement**: É a classe que representa um nó.
- **Classe SugiyamaLayout**: É a classe que representa a abordagem de Sugiyama, permitindo a execução de seu algoritmo de *layout* hierárquico.
- **Classe OptimalHierarchyLayout**: É a classe que representa o *layout* ótimo hierarquizado. Possui atributos para definição estética do desenho, tais como distâncias entre níveis, distâncias entre os nós, etc.

Um ponto importante nessa metodologia é que os desenhos gerados não são ortogonais, quando utilizadas as rotinas da OGDF. Por isso, um código simples foi utilizado para ortogonalizar as arestas resultantes deste método (usando-se o eixo médio da diagonal que forma a aresta e quebrando-a em partes horizontais e verticais.

Capítulo 6

Resultados

Neste capítulo serão apresentados os resultados de testes realizados para as implementações do método ortogonal baseado em topologia-forma-métrica, e do método hierárquico. Serão apresentados também resultados obtidos com dados do banco de informações dos circuitos da Light/RJ. Tais resultados demonstram o sistema que está funcionando na Light e já está operacional. O sistema Light é composto pelo gerador de sinóticos, código gerado na UFMG baseado neste trabalho de mestrado, rodando em uma máquina Sun (sistema operacional Solaris), e um sistema de visualização dos sinóticos gerados, implementado pela empresa Concert Technologies. A seguir, são mostrados resultados para validação dos sistemas. Primeiro, testes simples pontuais validam as restrições e o sistema linear como um todo, para as etapas da abordagem topologia-forma-métrica. Depois, testes preliminares com circuitos alimentadores da Light são mostrados. Após, é apresentada uma comparação entre abordagens, através dos resultados obtidos da metodologia hierárquica, para alguns circuitos alimentadores. As diferenças são discutidas e a escolha pela hierárquica é justificada. Por fim, é mostrada uma série de circuitos alimentadores desenhados automaticamente, pela abordagem definitiva. A visualização destes resultados neste trabalho são feitas por um aplicativo baseado em OpenGL (Team, 2008) desenvolvido para o propósito de se visualizar os desenhos em ambiente de laboratório (UFMG). Toda informação de desenho, geométrica, é exatamente a mesma informação do visualizador oficial do sistema Light, embora algumas informações específicas da caracterização (elétrica) dos elementos, que estão no programa visualizador da Concert, sejam ignoradas em laboratório.

6.1 Testes Simples Pontuais

Foram elaborados testes simples para conferir a correta montagem dos sistemas lineares das etapas de ortogonalização e compactação (nesse momento, é assumido que a planarização foi extensivamente testada (Melo, 2007)). Tais testes são compostos por combinações de grafos, pre-definidas em código fonte, que servirão de entrada para as rotinas de desenho. O que se fez, foi criar as relações de conexão entre vértices, formar o grafo e colocá-lo

como entrada para a montagem dos sistemas a serem resolvidos por programação linear inteira, conforme descrito no capítulo sobre a implementação da plataforma de desenhos. A figura 6.1 mostra alguns dos resultados obtidos.

É possível notar que o modelo Kandinsky com vértices de tamanho pré-determinado foi alvo de alguns dos testes. Neles, o vértice é explodido em um retângulo onde a altura, largura, e as posições de saídas das arestas podem ser controladas.

A figura 6.2 mostra um exemplo de controle de saída de arestas em um vértice explodido. Neste exemplo foram utilizadas as restrições que controlam, no vértice 1 o lado de saída da aresta. A aresta adjacente para o vértice 2 é colocada no lado oposto às arestas para os vértices 0, 3 e 4. A intenção era fazer com que as demais arestas partissem do lado oposto ao da aresta para o vértice 2, simulando o que poderia ser um exemplo de chave seccionadora a gás (vide capítulo 5). Com este exemplo é possível demonstrar o funcionamento de todas as rotinas implementadas para estruturação da etapa de ortogonalização e compactação. A figura 6.3 é uma extensão da figura 6.2 com o vértice principal tendo maior grau (maior número de arestas adjacentes). A partir dos testes, pontualmente foram estabelecidas todas as restrições dos sistema de ortogonalização e compactação.

6.2 Testes Preliminares com Circuitos Alimentadores da Light

Esta seção mostra os testes feitos com circuitos completos de alimentadores da Light. Para execução destes testes, foi utilizado o sistema total: leitura dos dados de entrada, montagem do grafo, planarização, ortogonalização PLI, compactação PLI, e escrita da saída.

6.2.1 Desempenho dos Testes Preliminares

Testes mostraram que a execução das rotinas de geração de desenho é bem rápida. A leitura do banco de dados para circuitos alimentadores longos (com número de elementos elevado) dura pelo menos 10 vezes mais tempo que a execução de todas as rotinas de geração e criação da saída. Com um tempo de execução entre dezenas de segundos e alguns minutos, este sistema atinge um patamar aceitável para execução. A tabela 6.1 mostra as médias de tempo de execução para os testes efetuados.

6.2.2 Resultados dos Testes Preliminares

As figuras 6.4 e 6.5 mostram desenhos de alimentadores. É possível notar que a compactação não atinge um resultado ótimo, o que é explicado pela escolha do método rápido, adotado na implementação. Neste exemplo, tal característica não afeta a visualização do alimentador em tela, por causa do número de elementos e pela disposição definida pela ortogonalização.

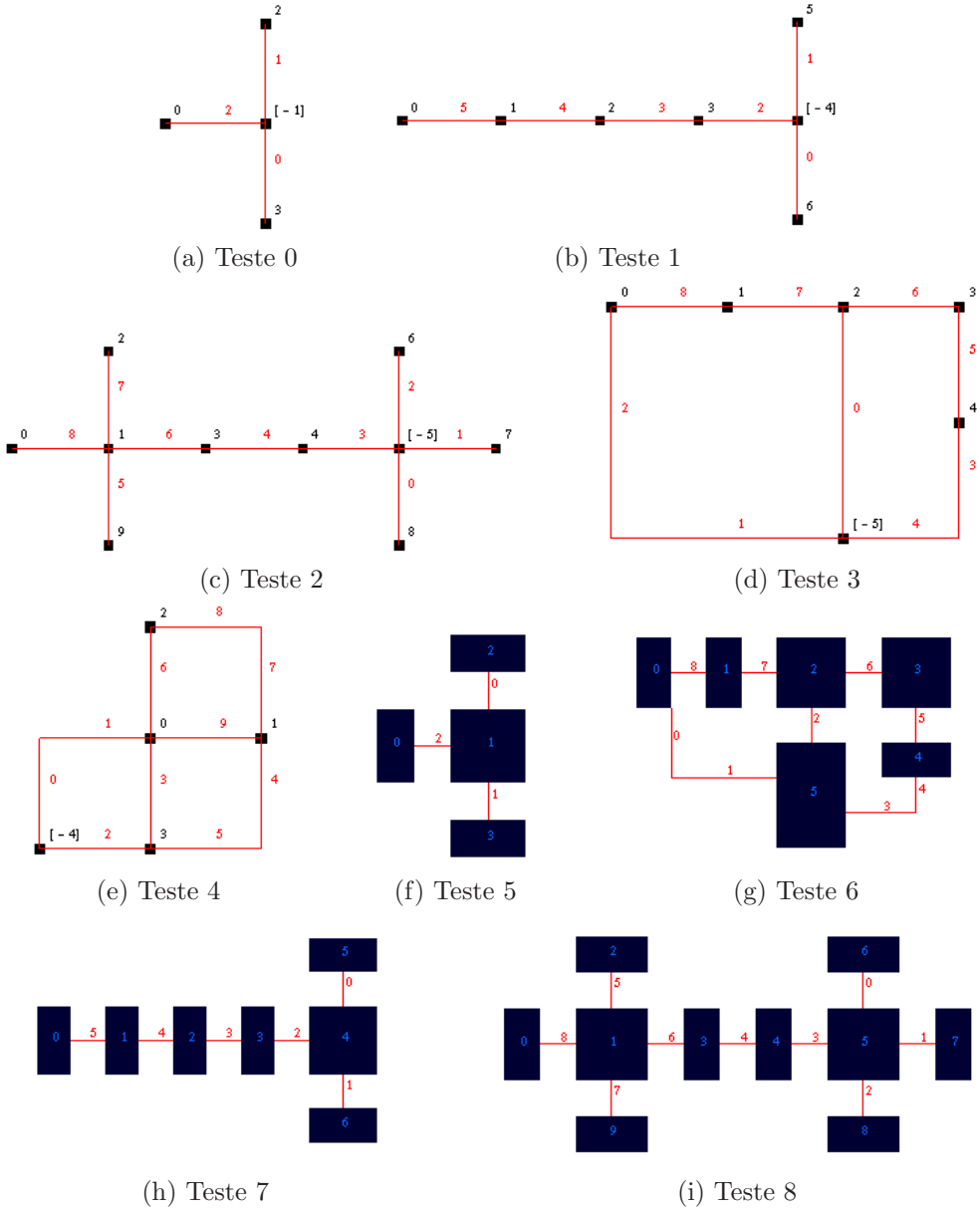
Tabela 6.1: Média de tempos de execução

	Duração dos Testes
Testes Simples	0.1s
Leitura banco de dados (menor que 10 elementos)	5s
Leitura banco de dados (entre 10 e 50 elementos)	45s
Leitura banco de dados (entre 50 e 100 elementos)	1m20s
Execução Sist. PLI (menor que 10 elementos)	1.2s
Execução Sist. PLI (entre 10 e 50 elementos)	4.7s
Execução Sist. PLI (entre 50 e 100 elementos)	15.0s
Execução Hierárquica (menor que 10 elementos)	0.2s
Execução Hierárquica (entre 10 e 50 elementos)	2.1s
Execução Hierárquica (entre 50 e 100 elementos)	5.8s

A figura 6.6 mostra outro alimentador, que possui um número maior de elementos. Neste resultado, é possível ver a melhor compactação em comparação com a figura 6.7 que mostra um sinótico gerado pelo antigo sistema da Light.

Uma característica presente nos resultados neste ponto foi a aparência *encaracolada* dos circuitos mais longos. Por causa de uma característica intrínseca do sistema de ortogonalização, baseado no sistema de fluxo máximo (Battista et al., 1999), vértices com grau maior que 2 sempre tendem a ter arestas que se direcionam para o centro do grafo, formando o *caracol* aparente. Alguns exemplos são apresentados nas figuras 6.9, 6.10 e 6.11. O ganho em termos de compactação, se comparados aos sinóticos anteriores ao sistema automático é bem aparente. Entretanto, este tipo de apresentação não é satisfatória com relação ao critério de legibilidade do sentido de fluxo de potência. Em uma apresentação de testes para engenheiros da Light, foi visto que a estruturação dos desenhos evidenciando a informação fluxo de potência seria bem-vinda.

Para atingir tal objetivo, mantendo-se os ganhos em compactação, eliminação de cruzamento de arestas, etc, foi estudada uma maneira de se controlar as arestas através de restrições na ortogonalização, eliminando-se o efeito "caracol". Entretanto, testes mostraram que a complexidade da programação linear aumentaria bastante com tais restrições, e, a abordagem baseada no Sugiyama (capítulo 4) poderia entregar desenhos com boa compactação, eliminação dos cruzamentos de arestas e uma hierarquia entre os elementos que ilustrava perfeitamente a idéia do fluxo de potência sendo consumido.



(a) Teste 0 (b) Teste 1

(c) Teste 2 (d) Teste 3

(e) Teste 4 (f) Teste 5 (g) Teste 6

(h) Teste 7 (i) Teste 8

Figura 6.1: Alguns dos resultados dos testes programados para ortogonalização e compactação PLI. (a) Teste 0: Teste mais simples, conferindo o correto posicionamento das arestas (ângulos de arestas) e seus comprimentos (que devem ser mínimos). (b) Teste 1: extensão do teste 0, conferindo comprimento das arestas e posicionamento em linha. (c) Teste 2: conferindo simetria (mesma resposta para setores idênticos). A menos que seja necessária uma resposta diferente (para se ter mais compactação por exemplo), é esperada uma mesma resposta para setores simétricos. (d) Teste 3: primeiro teste de face fechada, conferindo dobras de arestas. (e) Teste 4: outro teste de face fechada, conferindo dobras de arestas e posicionamento no vértice com grau 4. (f) Teste 5: teste mais simples (teste 0) agora com vértices explodidos. É conferido neste teste, toda configuração de criação dos vértices explodidos. (g) Teste 6: teste de faces fechadas com explosão de vértices. (h) Teste 7: replicação do teste 1 com vértices explodidos. (i) Teste 8: replicação do teste 2 com vértices explodidos.

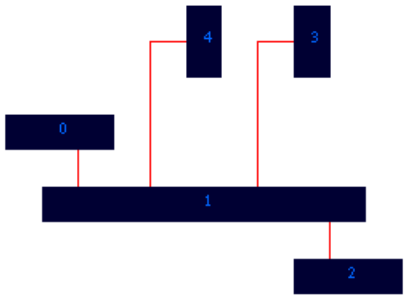


Figura 6.2: Teste para vértice com controle de saída de aresta.

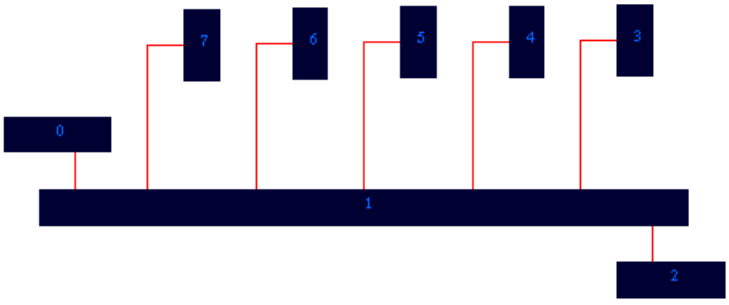


Figura 6.3: Teste para vértice com controle de saída de aresta, com várias arestas adjacentes.

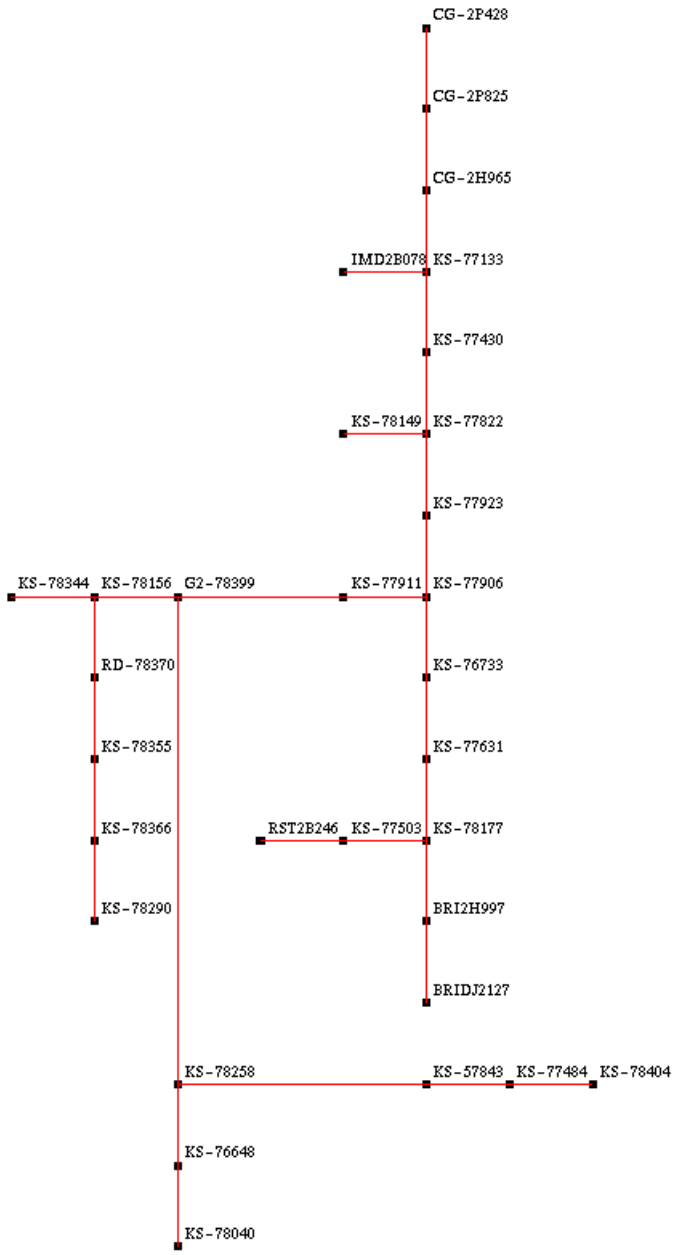


Figura 6.4: Teste com alimentador 1. Metodologia Topologia-Forma-Métrica.

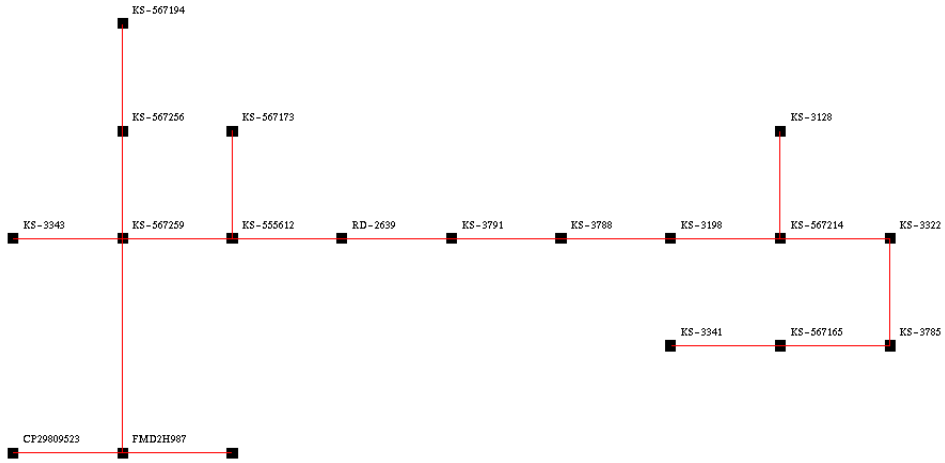


Figura 6.5: Teste com alimentador 2. Metodologia Topologia-Forma-Métrica.

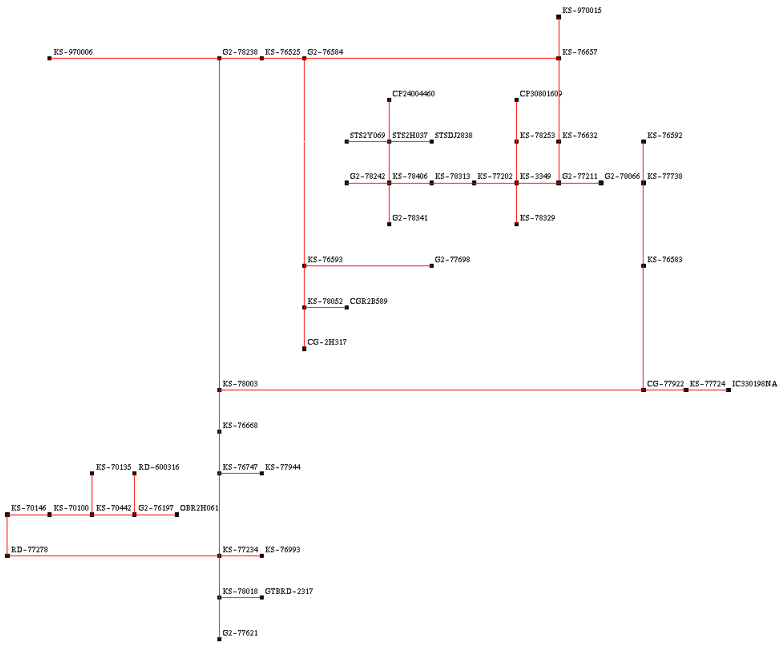


Figura 6.6: Teste com alimentador 3. Metodologia Topologia-Forma-Métrica.

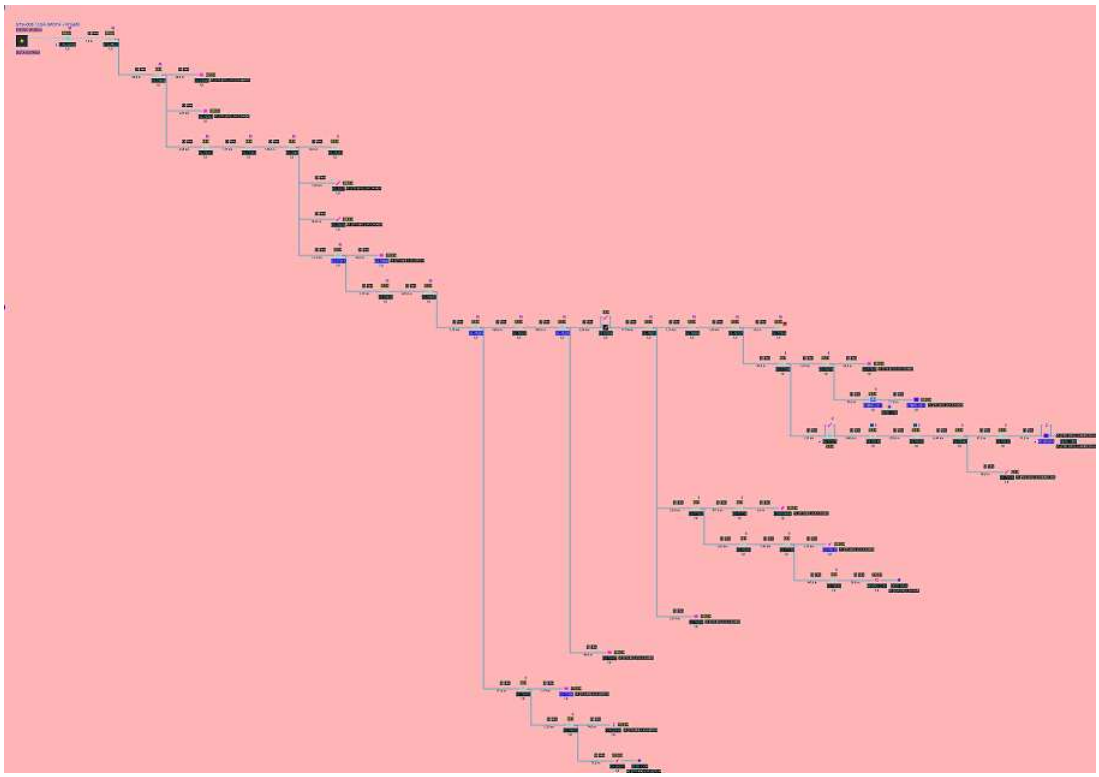


Figura 6.7: Sinótico gerado pelo antigo sistema da Light para o alimentador 3.

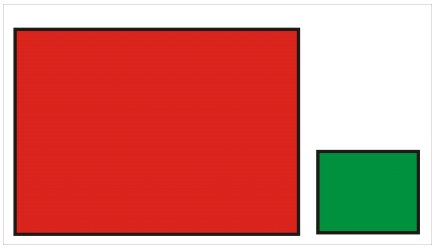


Figura 6.8: Esta figura mostra uma aproximação dos tamanhos dos desenhos para o alimentador 3. O retângulo vermelho se refere ao tamanho do alimentador desenhado no antigo sistema da empresa Light. O retângulo verde mostra o tamanho, em comparação ao sistema antigo, doa alimentador 3 desenhado no novo sistema, baseado na topologia-forma-métrica.

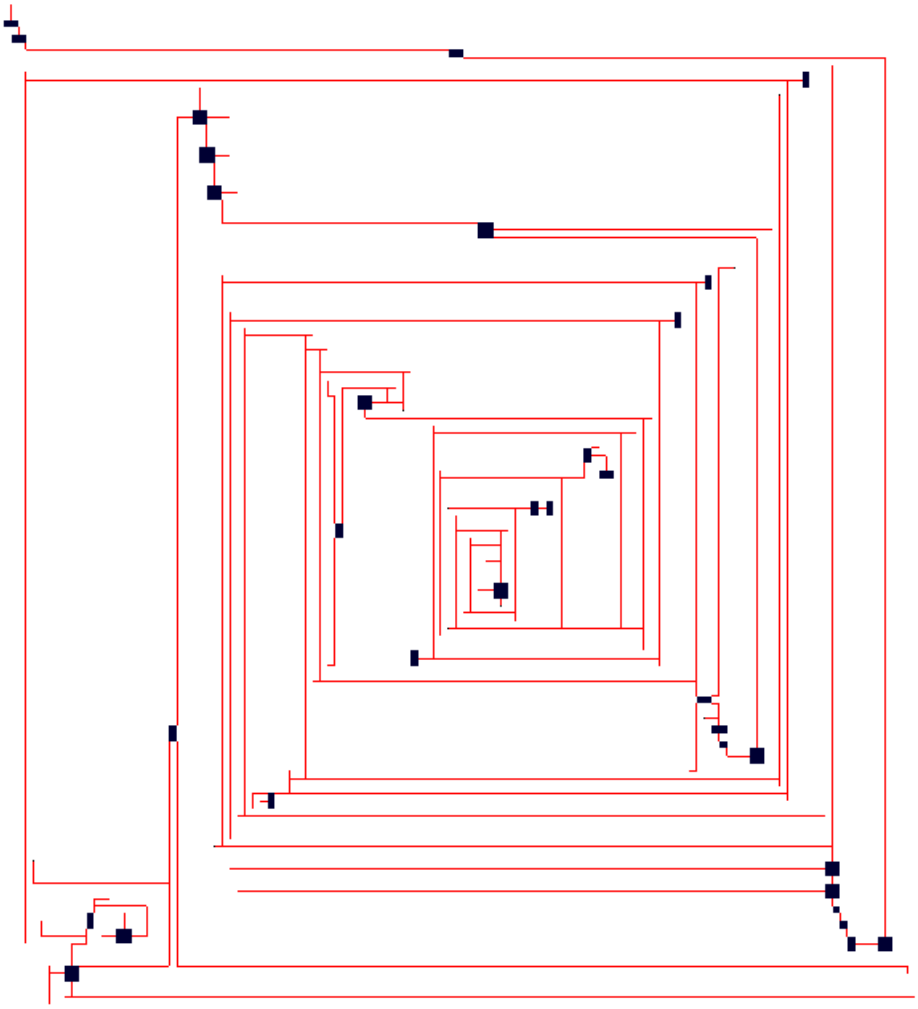


Figura 6.9: Teste para alimentador 4. Metodologia Topologia-Forma-Métrica, com efeito caracol.

6.3 Resultados Para Comparação entre Abordagens

Os resultados a seguir comparam os desenhos dos alimentadores mostrados nos testes preliminares seção 6.2 com os gerados pela metodologia hierárquica. A figura 6.12 mostra o alimentador 6 gerado baseado na abordagem hierárquica. É aparente a melhora em termos de legibilidade e noção de sentido do fluxo de energia, apesar de que a abordagem hierárquica gera desenhos com menor compactação. Mesma comparação é valida para o alimentador 4, mostrado nas figuras 6.13 e 6.9. A figura 6.14 mostra que, mesmo o resultado por estratégia hierárquica do alimentador 4 ser pior em compactação, comparado ao mesmo alimentador na estratégia topologia-forma-métrica, o resultado hierárquico ainda é mais compacto que o sistema antigo da empresa Light.

Estes resultados mostram que a abordagem hierárquica cumpre o critério de layout desejado (e de maneira mais simples que através de restrições na ortogonalização no modelo

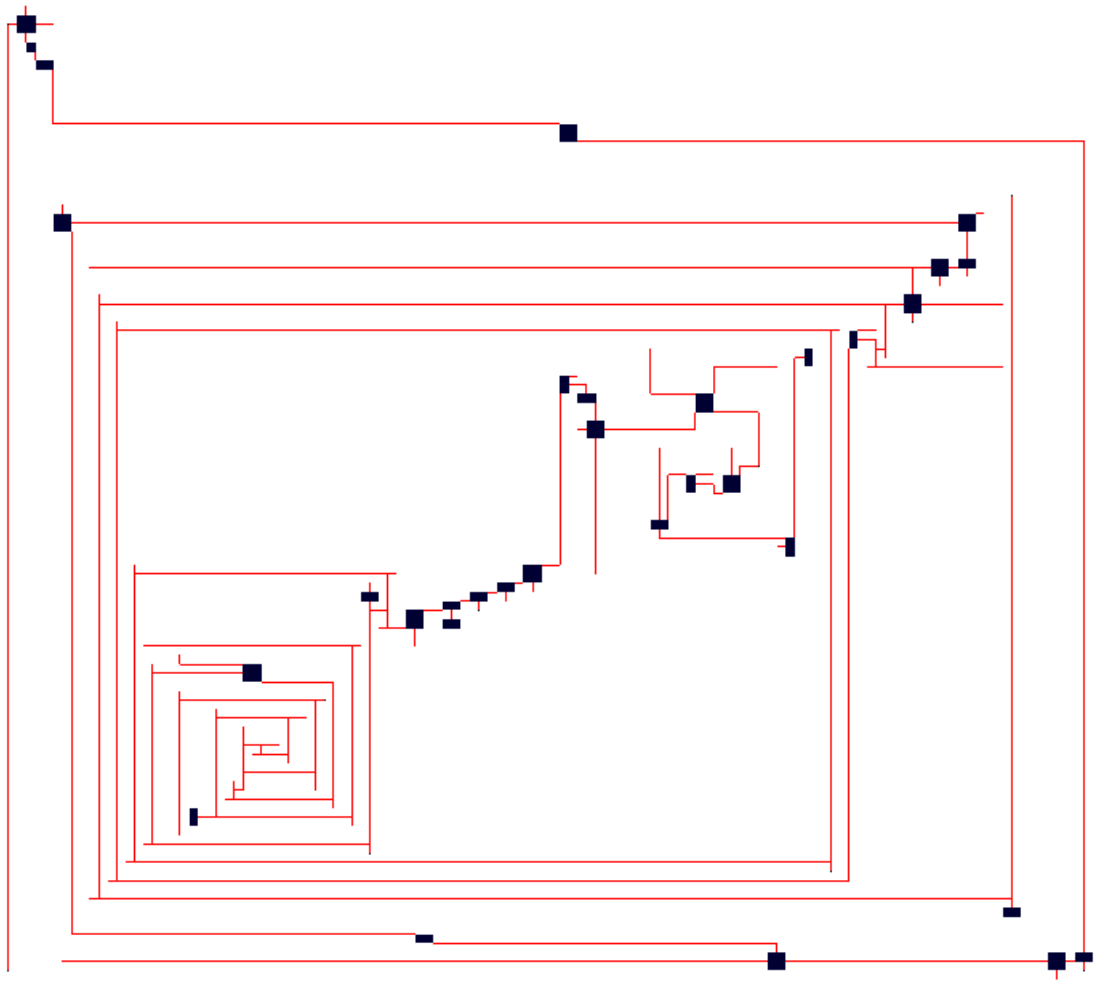


Figura 6.10: Teste para alimentador 5. Metodologia Topologia-Forma-Métrica, com efeito caracol e fluxo de potência não evidente.

topologia-forma-métrica). Apesar da menor compactação, como mostrado no resultado da figura 6.15, esta abordagem foi escolhida e aprovada para execução do sistema de geração de sinóticos ortogonais de circuitos alimentadores da Light.

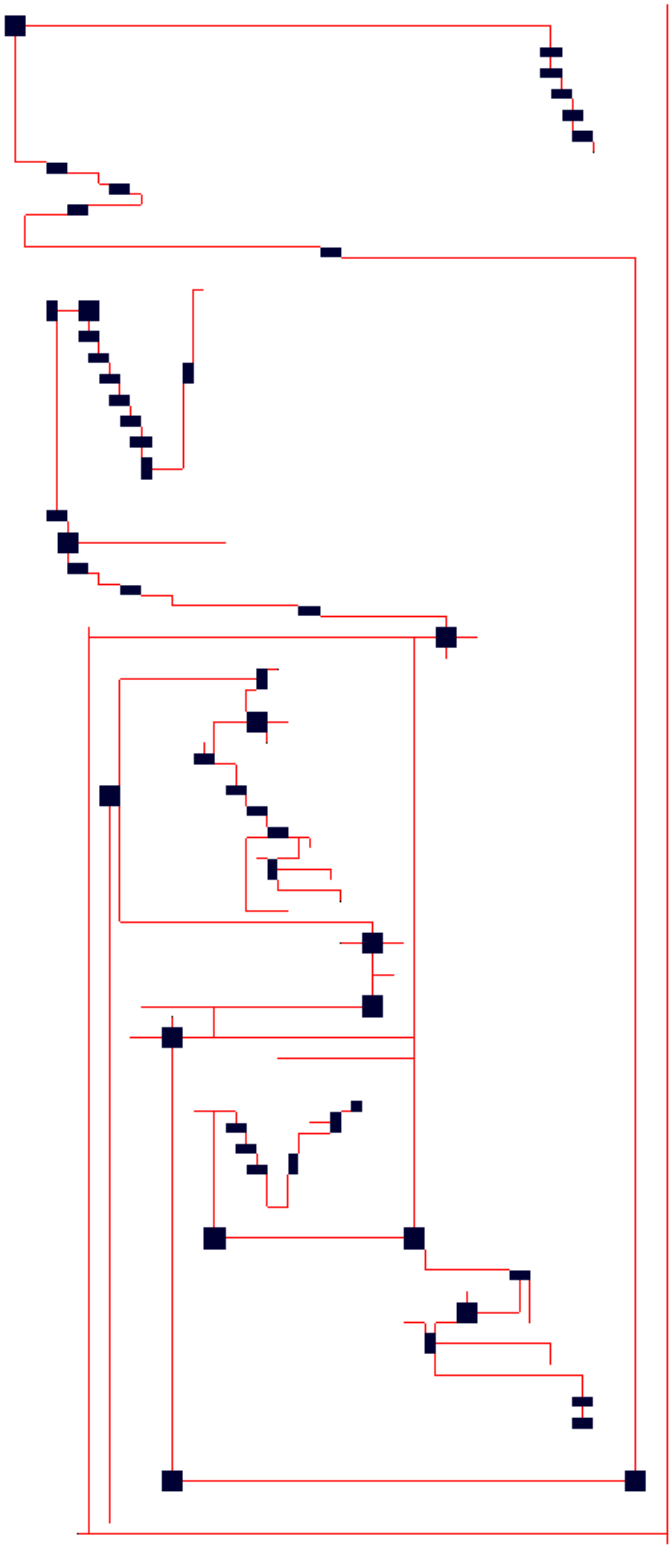


Figura 6.11: Teste para alimentador 6, usando a metodologia topologia-forma-métrica. Este desenho pode ser comparado ao resultado do teste do mesmo alimentador, sob metodologia hierárquica, mostrado na figura 6.12.

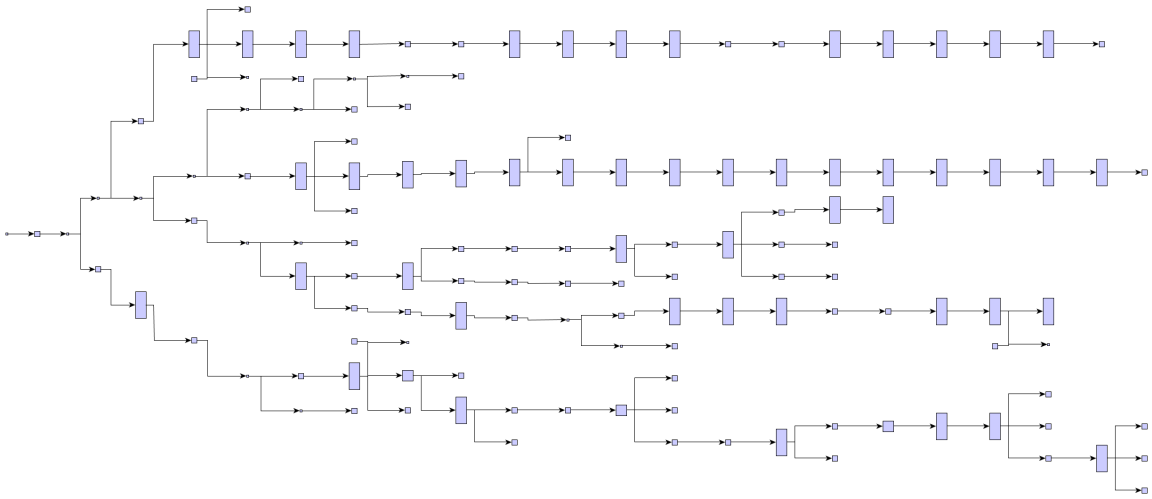


Figura 6.12: Teste para alimentador 6 usando a abordagem hierárquica.

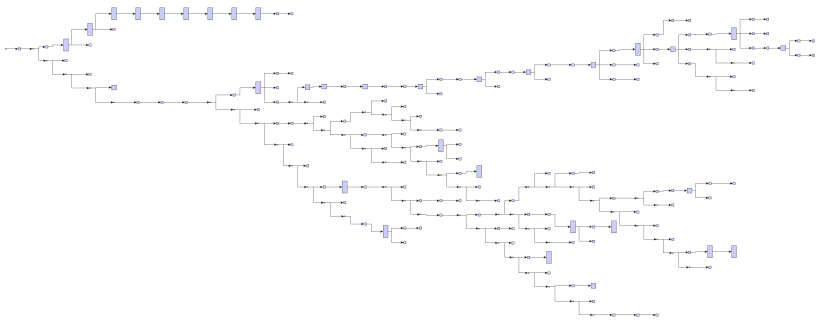


Figura 6.13: Teste para alimentador 4 usando a abordagem hierárquica. Compare com a figura 6.9.

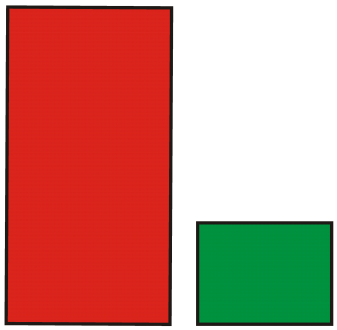


Figura 6.14: Nesta figura, o retângulo vermelho se refere ao tamanho do alimentador 4 desenhado no sistema antigo da Light. O retângulo verde, referente ao alimentador 4 desenhado por estratégia hierárquica, mostra que mesmo com resultados piores (em compactação) do que a topologia-forma-métrica, os desenhos hierárquicos conseguem obter grande compactação, pela eficiência em se posicionar os vértices nos níveis.

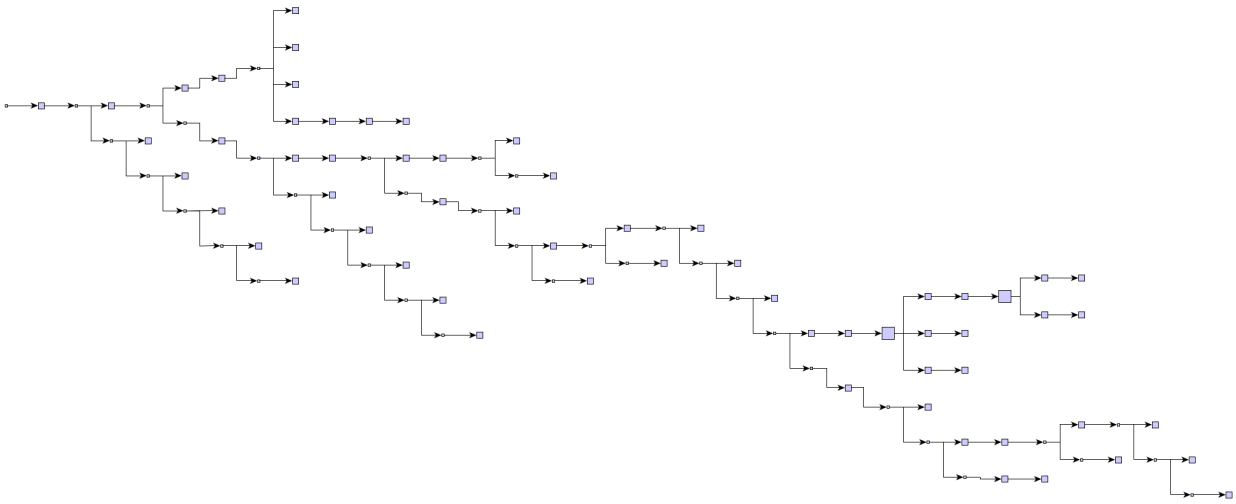


Figura 6.15: Teste do alimentador 7 usando a abordagem hierárquica.

6.4 Resultados dos Testes Definitivos

A partir da figura 6.16, são apresentados desenhos com resultados obtidos do sistema de geração de sinóticos ortogonais automáticos, com dados dos circuitos alimentadores da Light. Tais resultados, colocados como definitivos, seguem a implementação feita sob a abordagem hierárquica e mostra circuitos alimentadores reais. O sistema completo (gerador e visualizador) gera os esquemáticos a partir da base de dados completa, e testes de operação real, feitos pela equipe da Light, homologaram o sistema.

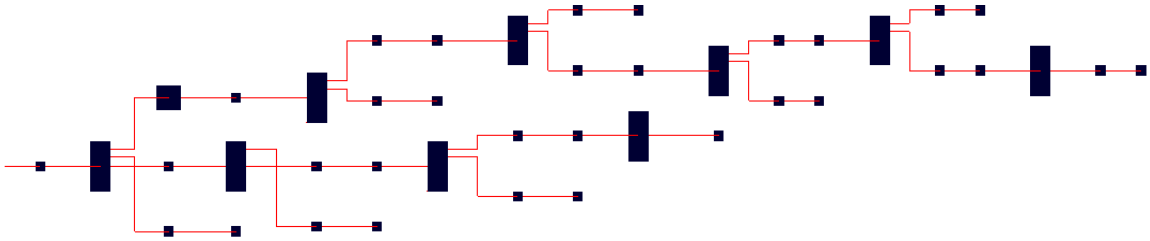


Figura 6.16: Este resultado mostra a diferenciação entre os tipos de vértices especiais. A abordagem hierárquica permite a leitura de tal informação de maneira bem simplificada.

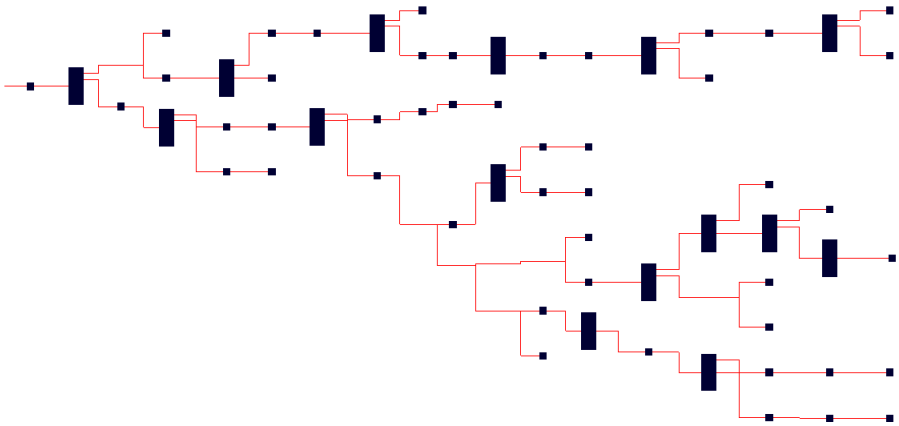


Figura 6.17: Outro resultado com diferenciação clara de elementos especiais (vértices).

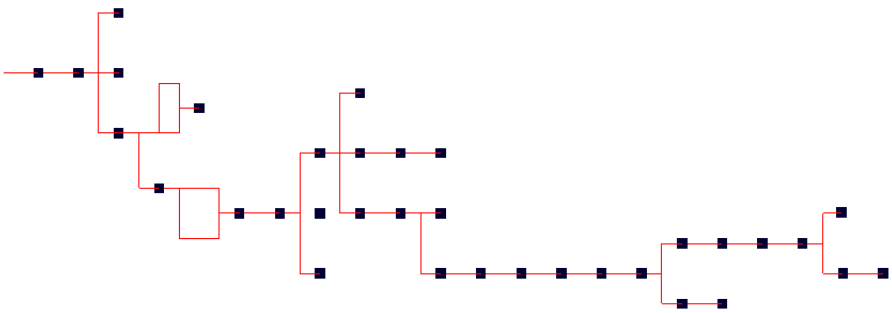


Figura 6.18: Este resultado mostra como ciclos são representados nos diagramas.

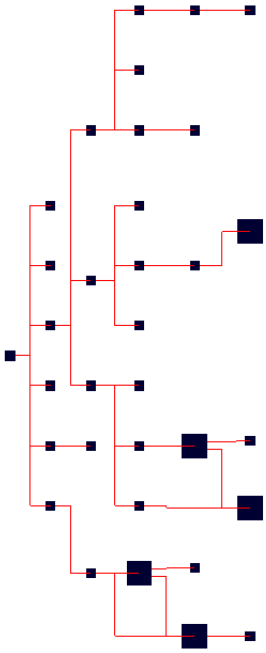


Figura 6.19: Outro resultado com exemplos de ciclos mapeados pelo método.

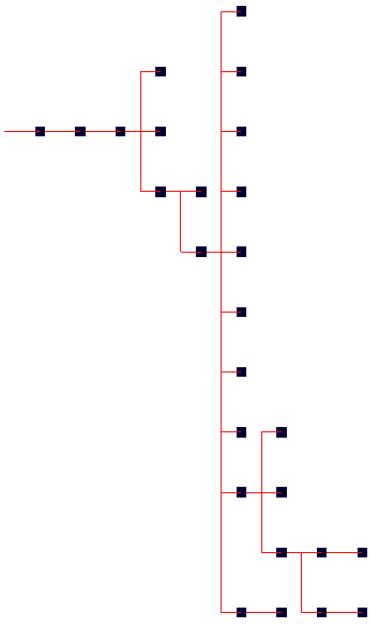


Figura 6.20: Apesar da baixa compactação, a informação de elementos em mesmo nível é destacada para os circuitos de alimentadores. Tal informação é relevante ao se definir grupos de elementos interligados a uma fonte comum.

Capítulo 7

Conclusões

Sistemas geradores de desenhos automáticos de grafos são considerados de vital importância quando existe a necessidade de visualizar dados com clareza, boa legibilidade e com rapidez. No setor elétrico, os sinóticos são peças importantes para os operadores que necessitam de uma resposta visual do seu sistema com qualidade e eficiência.

Este trabalho propôs desenvolver um sistema de geração de esquemáticos ortogonais. Para atingir tal objetivo, foram desenvolvidas duas abordagens destacadas pela possibilidade de flexibilização da solução. Para isso, primeiramente o problema de desenho de grafos foi modelado utilizando programação linear inteira. Através da PLI pôde-se obter uma ferramenta genérica que possibilita a inserção de qualquer critério ou requisito de desenho, modelando a especificidade do problema. Foi visto, neste estudo, que os algoritmos Giotto e Kandinsky são os mais eficientes em termos de vários critérios estéticos. A utilização da abordagem PLI se justifica pela necessidade de se obter generalidade na construção dos critérios estéticos. Desta forma, o contexto da aplicação de desenho de esquemáticos de circuitos elétricos pode ser amplamente coberto. A abordagem por PLI fornece uma solução exata porém, é limitada a problemas de tamanho pequeno a médio. Para grafos muito grandes ou envolvendo muitos nós, heurísticas tornam-se mais interessantes.

A abordagem topologia-forma-métrica possibilitou, através da PLI, que algumas restrições de desenhos específicas pudessem ser modeladas. Entretanto, essa abordagem é complexa quando comparada com a hierárquica, especificamente para o desenho dos esquemáticos deste estudo. Isso porque um dos critérios mais importantes definidos como requisitos da solução seria a montagem dos desenhos de grafos em uma forma hierárquica, evidenciando a noção de distribuição de fluxo de potência, entre os elementos elétricos mostrados. Para implementar tal critério sob moldes do Kandisky (Eiglsperger et al., 2000), era preciso utilizar algumas restrições que causariam redução do espaço de soluções factíveis. Mesmo com resultados melhores com relação à compactação dos desenhos, o Kandinsky foi substituído pelo método hierárquico Sugiyama (Sugiyama, 2002) especialmente desenvolvido para desenhos com este critério. No contexto de aplicação de desenho dos circuitos deste estudo, esta abordagem gerou resultados mais próximos dos esperados pelos usuários

do sistema.

Resultados reais obtidos em um sistema rodando dados de alimentadores de uma concessionária de energia do Rio de Janeiro foram gerados. Uma plataforma geradora de esquemáticos foi desenvolvida para ler dados de tais circuitos e montar grafos que deveriam ser mostrados em uma tela de sinótico. O sistema funciona como visualizador de esquemáticos ortogonais da Light.

Além de servir ao propósito original, como sistema de geração de esquemáticos para Light, este trabalho também possibilitou definir a base de estudos para melhorias em desenhos de grafos, baseadas em restrições para etapas de ortogonalização e compactação dos desenhos (sob a metodologia topologia-forma-métrica). Estudos presentes buscam adequar, primeiramente, a noção de razão de aspecto aos desenhos, na etapa de compactação através de sistemas multi-objetivos.

Como trabalhos futuros, propõe-se trabalhar com novas restrições do sistema PLI para modelar novos critérios/requisitos de desenhos. Além disso é necessário que se trabalhe na compactação, para que resultados melhores que a metodologia rápida apresentada possam ser obtidos, trabalhando-se nas heurísticas propostas por Eiglsperger and Kaufmann (2001).

Referências Bibliográficas

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, New Jersey, USA, first edition. [citado na(s) páginas(s) 19]

Barth, W., Jünger, M., and Mutzel, P. (2002). Simple and efficient bilayer cross counting. *Proceedings of the Symposium on Graph Drawing of LNCS*, 2528:130–141. [citado na(s) páginas(s) 49]

Batine, C., Furlani, L., and Nardelli, E. (1985). What's a good diagram? A pragmatic approach. In *In Proc. 4th Internat. Conf. on the Entity Relationship Approach*. [citado na(s) páginas(s) 10]

Batini, C., Nardelli, E., and Tamassia, R. (1986). A layout algorithm for data-flow diagrams. *IEEE Trans. Softw. Eng.*, 4:538–546. [citado na(s) páginas(s) 15, 16, 33]

Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. (1994). Algorithms for drawing graphs: An annotated bibliography. Technical report, Brown University. [citado na(s) páginas(s) 1, 2, 15]

Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. (1999). *Graph Drawing - Algorithms for the visualization of graphs*. Prentice-Hall. [citado na(s) páginas(s) 1, 5, 8, 11, 14, 15, 17, 49, 60, 65]

Battista, G. D., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., and Vaugir, F. (1995). An experimental comparison of four graph drawing algorithms. *11th Annual ACM Symposium on Computational Geometry*. [citado na(s) páginas(s) 25]

Battista, G. D. and Tamassia, R. (1988). Algorithms for plane representation of acyclic digraphs. *Theoret. Comput. Sci.*, (61):175–198. [citado na(s) páginas(s) 15, 23]

Battista, G. D., Tamassia, R., and Tollis, I. G. (1992). Constrained visibility representations of graphs. *Inform. Process. Lett.*, (41):1–7. [citado na(s) páginas(s) 15]

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, (6):87–90. [citado na(s) páginas(s) 17]

Bertolazzi, P., Coehn, R. F., Battista, G. D., Tamassia, R., and Tollis, I. G. (1994). How to draw a series-parallel digraph. *International Journal of Computational Geometry and Applications*, 4(4):385–402. [citado na(s) páginas(s) 15]

Biggs, N. L., Lloyd, K., and Wilson, R. (1976). *Graph Theory*. Oxford University Press, England. [citado na(s) páginas(s) 5]

Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. The Macmillan Press. [citado na(s) páginas(s) 5, 16]

Brandes, U. and Köpf, B. (2002). Fast and simple horizontal coordinate assignment. *Proceedings of the Symposium on Graph Drawing of LNCS*, 2265(31-44). [citado na(s) páginas(s) 50]

Carpano, M. J. (1980). Automatic display of hierarchized graphs for computer-aided decision analysis. *IEEE Trans. Syst. Man Cybern.*, SMC-10(11):705–715. [citado na(s) páginas(s) 15]

Chimani, M., Gutwenger, C., Jünger, M., Klein, K., Mutzel, P., and Schulz, M. (2007). The open graph drawing framework. 15th International Symposium on Graph Drawing. [citado na(s) páginas(s) 61]

Chvatal, V. (1983). *Linear Programming*. Freeman. [citado na(s) páginas(s) 25]

Coffman Jr., E. G. and Graham, R. L. (1972). Optimal scheduling for two-processor systems. *Acta Informatica*, 1(3):200–213. [citado na(s) páginas(s) 47]

Deb, K. (2003). *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley and Son. [citado na(s) páginas(s) 11]

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, (1):269–271. [citado na(s) páginas(s) 17]

Eades, P. and Wormald, N. C. (1994). Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403. [citado na(s) páginas(s) 47, 49]

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, second edition. [citado na(s) páginas(s) 11]

Eiglsperger, M., Fekete, S. P., and Klau, G. W. (2001). Orthogonal graph drawing. In *Lecture Notes in Computer Science*, number 2025, pages 121–171. [citado na(s) páginas(s) 16]

Eiglsperger, M., Fößmeier, U., and Kaufmann, M. (2000). Orthogonal graph drawing with constraints. In *In Proceedings of the eleventh annual ACM-SIAM*, pages 3–11. Symposium on discrete algorithms. [citado na(s) páginas(s) 16, 25, 26, 27, 30, 31, 33, 34, 59, 79]

Eiglsperger, M. and Kaufmann, M. (2001). Fast compaction for orthogonal drawings with vertices of prescribed size. In Springer, editor, *In Proceedings of the 9th International Symposium on Graph Drawing (GD'2001)*, volume 2265, pages 124–138. [citado na(s) páginas(s) 16, 26, 36, 40, 41, 42, 43, 80]

Eiglsperger, M., Kaufmann, M., and Siebenhaller, M. (2003). A topology-shape-metrics approach for the automatic layout of UML class diagrams. In Press, A., editor, *In Proceedings of the 2003 ACM Symposium on Software Visualization*, New York, NY. [citado na(s) páginas(s) 33]

Eiglsperger, M., Siebenhaller, M., and Kaufmann, M. (2005). An efficient implementation of sugiyama's algorithm for layered graph drawing. *Journal of Graph Algorithms and Applications*, 9(3):305–325. [citado na(s) páginas(s) 23, 48]

Fößmeier, U. and Kaufmann, M. (1994). On bend-minimum orthogonal upward drawing of directed planar graphs. *Proceedings of Graph Drawings*, LNCS 894:52–63. [citado na(s) páginas(s) 30]

Fößmeier, U. and Kaufmann, M. (1996). Drawing high degree graphs with low bend numbers. *In Proceedings of Graph Drawing*, 1027:254–266. [citado na(s) páginas(s) 2, 17, 25, 33]

Fößmeier, U. and Kaufmann, M. (1997). Algorithms and area bounds for nonplanar orthogonal drawings. *Proceedings of Graph Drawing*, 1353:134–145. [citado na(s) páginas(s) 2, 25]

Ford, L. R. and Fulkerson, D. R. (1962). *Flows in networks*. Princeton University Press. [citado na(s) páginas(s) 17]

Foulds, L. R. (1992). *Graph Theory Applications*. Springer-Verlag, New York, first edition. [citado na(s) páginas(s) 21]

Frick, A. (1997). Upper bounds on the number of hidden nodes in sugiyama's algorithm. *Proceedings of the Symposium on Graph Drawing of LNCS*, 9(3):169–183. [citado na(s) páginas(s) 48, 51]

Gansner, E., Koutsofios, E., North, S., and Vo, K. P. (1993). A technique for drawing directed graphs. *IEEE Transaction. Software Engeneering*, 3(19):214–230. [citado na(s) páginas(s) 48, 50]

Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1974). Some simplified NP-complete problems. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, New York, NY, USA. ACM. [citado na(s) páginas(s) 47]

Garg, A. and Tamassia, R. (1994). On the computational complexity of upward and rectilinear planarity testing. *Proceedings of the DIMACS Workshop of Graph Drawing*, LNCS 894:186–197. [citado na(s) páginas(s) 30]

Gross, J. and Yellen, J. (1999). *Graph Theory and its Applications*. CRC Pres. [citado na(s) páginas(s) 5]

Harary, F. (2009). *Graph Theory*. Addison-Wesley. [citado na(s) páginas(s) 5]

Healy, P. and Nikolov, N. (2002). How to layer a directed acyclic graph. *Proceedings of the Symposium on Graph Drawing of LNCS*, 2265:16–30. [citado na(s) páginas(s) 48]

Indra (2007). CS2007-00814 novo esquemático ortogonal, especificação técnica. [citado na(s) páginas(s) 58]

Jayakumar, R., Thulasiraman, K., and Swamy, M. N. (1986). An optimal algorithm for maximal planarization of nonplanar graphs. *IEEE Internat. Sympos. on Circuits and Systems*, pages 1237–1240. [citado na(s) páginas(s) 17]

Jünger, M. and Mutzel, P. (1997). Maximum planar subgraphs an nice embeddings: Practical layout tools. *Proc. Graph Drawing*, pages 193–204. [citado na(s) páginas(s) 17]

Kant, G. (1992). An $O(n^2)$ maximal planarization algorithm based on pq-trees. Technical report, Dept. Comput. Sci. Utrecht Univ. [citado na(s) páginas(s) 17]

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press. [citado na(s) páginas(s) 47]

Klau, G. W. and Mutzel, P. (1999a). Combining graph labeling and compaction (extended abstract). In *Eds. Proceedings Graph Drawing*, pages 22–37. [citado na(s) páginas(s) 16]

Klau, G. W. and Mutzel, P. (1999b). Optimal compaction of orthogonal grid drawings. *In Integer Programming and Combinatorial Optimization (IPCO'99)*, (1610):304–319. [citado na(s) páginas(s) 16, 36, 40, 59]

Klau, G. W. A. (2001). *A Combinatorial Approach to Orthogonal Placement Problems*. PhD thesis, Universität des Saarlandes. [citado na(s) páginas(s) 16]

Kosak, C., Marks, J., and Shieber, S. (1994). Automating the layout of network diagramas with specified visual organization. *IEEE Trans. Syst. Man Cybern.*, SMC-11(3):440–454. [citado na(s) páginas(s) 13]

Lengauer, T. (1990). *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley and Sons. [citado na(s) páginas(s) 36]

Lipton, R. J., North, S. C., and Sandberg, J. S. (1985). A method for drawing graphs. In *In Proc. 1st Annual ACM Symposium Computational Geometry*, pages 153–160. [citado na(s) páginas(s) 11]

Melo, L. T. C. (2007). Uma biblioteca para desenho de grafos, construída sobre o paradigma de programação genérica. Master's thesis, Universidade Federal de Minas Gerais. [citado na(s) páginas(s) 2, 11, 60, 63]

Mesquita, R. C. and Terra, F. M. (2007). Implementações das ferramentas gráficas de software - entrega código fonte c++, relatório 4 do projeto P&D 008/06 light (aneel/concert/ufmg) - desenvolvimento de ferramenta para geração automática de diagrama ortogonal das redes de distribuição. [citado na(s) páginas(s) 59]

Nardelli, E. and Talamo, M. (1984). A fast algorithm for planarization of sparse diagrams. Technical report, IASI-CNR. [citado na(s) páginas(s) 17]

Ong, Y. S., Gooi, H. B., and Chan, C. K. (2000). Algorithms for automatic generation of one-line diagrams. *IEE Proceedings Generation, Transmission and Distribution*, 147(5):292–298. [citado na(s) páginas(s) 3]

Papadimitriou, C. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall. [citado na(s) páginas(s) 25]

Papakostas, A. and Tollis, I. G. (1994). Improved algorithms and bounds for orthogonal drawings. *Graph Drawing (Proc. GD '94)*. [citado na(s) páginas(s) 2]

Paterson, J. L. (1977). Petri nets. *ACM Computing Surveys*, 9(3):223–252. [citado na(s) páginas(s) 1]

Patrignani, M. (1999). On the complexity of orthogonal compaction. Technical report, Dipartimento di Informatica e Automazione, Università degli Studi di Roma Tre. [citado na(s) páginas(s) 36, 38]

Purchase, H. C. and Cohen, R. F. (1995). Validating graph drawing aesthetics. In Sci., L. N. C., editor, *Graph Drawing (Proc. GD '95)*, number 1027, pages 435–446. [citado na(s) páginas(s) 10]

Rao, P. S. N. and Deekshit, R. (2003). Distribution feeder one-line diagram generation: a visibility representation. *Electric Power Systems Research*, 70(3):173–178. [citado na(s) páginas(s) 3]

Reingold, E. and Tilford, J. (1981). Tidier drawing of trees. *IEEE Transactions Softw. Eng.*, SE-7(2):223–228. [citado na(s) páginas(s) 15, 54]

Sander, G. (1999). Graph layout for applications in compiler construction. *Theoretical Computer Sciences*, 2(217):175–214. [citado na(s) páginas(s) 48, 49, 50]

Sedgewick, R. (2002). *Algorithms in C++ - Graph Algorithms*. Addison-Wesley, USA, third edition. [citado na(s) páginas(s) 21]

Soukup, J. (1972). Circuit layout. *Proc. IEEE*, 69(10):197–213. [citado na(s) páginas(s) 15]

Sugiyama, K. (2002). *Graph Drawing and Applications for Software and Knowledge Engineers*. World Scientific. [citado na(s) páginas(s) 23, 45, 59, 61, 79]

Sugiyama, K., Tagawa, S., and Toda, M. (1981). Methods for visual understanding of hierarchical systems. *IEEE Trans. Syst. Man Cybern.*, (2):109–125. [citado na(s) páginas(s) 3, 10, 15, 23, 24, 45, 49]

Tamassia, R. (1985). New layout techniques for entity-relationship diagrams. In *In Proc. 4th Internat. Conf. on Entity-Relationship Approach*, pages 304–311. [citado na(s) páginas(s) 17]

Tamassia, R. (1987). On embedding a graph in the grid with the minimum number of bends. *SIAM Journal Comput.*, (3):421–444. [citado na(s) páginas(s) 2, 15, 17, 18, 25, 27, 33, 36, 40]

Tamassia, R., Battista, G. D., and Batini, C. (1988). Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern*, (1):61–79. [citado na(s) páginas(s) 13, 15, 17, 25]

T.Biedl and Kant, G. (1998). A better heuristic for orthogonal graph drawings. *In Proc. 2nd Annu. European Sympo. Algorithms (ESA '94)*. [citado na(s) páginas(s) 2]

Team, L. (2003). LPSOLVE v. 5.5.0.10. http://lpsolve.sourceforge.net/5.5/. [citado na(s) páginas(s) 60]

Team, O. (2008). Opengl - the industry's foundation for high performance graphics. http://www.opengl.com/. [citado na(s) páginas(s) 63]

Waddle, V. and Malhortra, A. (1999). An $Elog(E)$ line crossing algorithm for leveled graphs. *Proceedings of the Symposium on Graph Drawing of LNCS*, 1731:59–71. [citado na(s) páginas(s) 49]

Walker, J. Q. (1990). A node-positioning algorithm for general trees. *Softw. Pract. Exp.*, 20(7):685–705. [citado na(s) páginas(s) 15, 54]

Warfield, J. (1977). Crossing theory and hierarchy mapping. *IEEE Trans. Systems, Man and Cybernetics*, SMC-7(7):502–523. [citado na(s) páginas(s) 15]

West, D. (2006). *Introduction to Graph Theory*. Prentice Hall, third edition. [citado na(s) páginas(s) 5]

Wilson, R. (1996). *Introduction to Graph Theory*. Longman, England, fourth edition. [citado na(s) páginas(s) 5, 17]