UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE NOVAS METODOLOGIAS PARA DESENHO AUTOMÁTICO DE GRAFOS BASEADAS EM OTIMIZAÇÃO

AUTORA: BERNADETE MARIA DE MENDONÇA NETA
ORIENTADORES:
PROF. DR. RENATO CARDOSO MESQUITA
PROF. DR. FREDERICO GADELHA GUIMARÃES

BELO HORIZONTE, MG
Dezembro 2010

Bernadete Maria De Mendonça Neta

Desenvolvimento de Novas Metodologias Para Desenho Automático de Grafos baseadas em Otimização

Orientação:
Prof. Dr. Renato Cardoso Mesquita
Prof. Dr. Frederico Gadelha Guimarães

Tese 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 a obtenção do grau de Doutor em Engenharia Elétrica.

Universidade Federal de Minas Gerais
Belo Horizonte, MG
Dezembro 2010

DEDICATÓRIA

Dedico este trabalho a minha mãe, Odete, pelo exemplo de conduta e honestidade e pelo apoio a mim dedicado, estando sempre disposta a me auxiliar no que fosse preciso.

Ao meu esposo, Rogério Braga, pelo incentivo e paciência durante esta caminhada.

Aos meus filhos, Ana Paula e Pedro Henrique, pela compreensão, carinho e solidariedade sempre.

Aos meus irmãos, sobrinhos e cunhados pelos constantes incentivos.

AGRADECIMENTOS

À CEMIG, pela disponibilização do horário e a importante oportunidade de desenvolvimento e aplicação desse trabalho. Em especial a Maria Helena Barbosa, Arnoldo Magela Morais, Ricardo L. J. Carnevalli, Rita de Cássia G. Fajardo e Luis Antonio Felber.

Aos orientadores: Professor Dr. Renato Cardoso Mesquita e Professor Dr. Frederico Gadelha Guimarães, pela orientação, atenção, paciência e carinho com que conduziram e orientaram este trabalho. Agradeço à grandiosa e oportuna colaboração e demonstro aqui, com estas palavras, o meu mais profundo respeito, carinho e admiração.

Aos colaboradores e professores do PPGEE - Programa de Pós-Graduação em Engenharia Elétrica da UFMG, em especial a Anete Vidal de Freitas Vieira, Arlete Vidal de Freitas, ao professor Hani Camille Yehia e ao professor Guilherme Augusto Silva Pereira, pelo absoluto apoio e incentivo no momento em que pareceu não mais ser possível continuar. Este apoio foi decisivo para o sucesso nessa jornada.

Ao Professor Oriane Magela Neto que, com sabedoria, inteligência e dedicação torna a ação de ensinar, uma verdadeira obra de arte.

Ao Professor Petr Iakovlevitch Ekel por todos os sábios ensinamentos, pela simplicidade, bem como pela sincera amizade. Demonstro aqui o meu mais profundo carinho, respeito e eterna admiração.

Ao Gustavo H. D. Araujo pela dedicação e auxilio nos desenvolvimentos computacionais, bem como por todas as contribuições durante o desenvolvimento deste trabalho.

Aos colegas da CEMIG que direta ou indiretamente contribuíram para a realização desse trabalho.

Ao meu esposo Rogério Braga, pelo apoio, carinho, atenção e paciência durante minha dedicação aos estudos.

Ao meu querido filho Pedro Henrique pela compreensão quando da falta de um colo.

À minha querida filha Ana Paula pela ajuda incondicional em todas as fases de desenvolvimento desse trabalho e ao meu genro Érico que da mesma forma, não economizou esforços em me ajudar no que foi preciso.

À minha querida mãe Odete, aos meus irmãos: Mª. De Lourdes, Mª. Aparecida, Mª. Luiza, e Ananias, e sobrinhas: Ingrid, Pollyana e Dayanne. Agradeço a todos por sempre me apoiarem e me conduzirem aos bons caminhos da vida.

À família de meu marido pelos constantes incentivos e compreensão por minhas constantes ausências.

Ao meu querido amigo Miguel Lima Mendes pelo apoio nos assuntos técnicos de maneira geral, assim como pela sincera amizade.

Ao meu querido amigo Agnaldo Antonio de Souza pelo apoio, incentivos constantes e a alegria contagiante. Da mesma forma a todos os amigos que me acompanharam pacientemente nesta árdua jornada.

A todos aqueles que de maneira direta ou indiretamente contribuíram para viabilização desse trabalho.

Agradeço, em especial, a **DEUS** que está sempre ao meu lado nos momentos tranquilos e me carrega em seus braços nos momentos mais difíceis de minha vida.

HOMENAGEM

A Fernando Sabino

A Última Crônica

"(...) A perspectiva me assusta. Gostaria de estar inspirado, de coroar com êxito mais um ano nesta busca do pitoresco ou do irrisório no cotidiano de cada um. Eu pretendia apenas recolher da vida diária algo de seu disperso conteúdo humano, fruto da convivência, que a faz mais digna de ser vivida. Visava ao circunstancial, ao episódico. Nesta perseguição do acidental, quer num flagrante de esquina, quer nas palavras de uma criança ou num acidente doméstico, torno-me simples espectador e perco a noção do essencial. Sem mais nada para contar, curvo a cabeça e tomo meu café, enquanto o verso do poeta se repete na lembrança: "assim eu quereria o meu último poema". Não sou poeta e estou sem assunto. Lanço então um último olhar fora de mim, onde vivem os assuntos que merecem uma crônica.(...)"

Fernando Sabino

RESUMO

Este trabalho tem como objetivo geral o estudo e desenvolvimento de novas metodologias para desenho automático de grafos. Estas visam auxiliar na resolução de importantes problemas relacionados com a qualidade, legibilidade, confiabilidade e visibilidade das informações providas por aplicativos que utilizam recursos relacionados com a representação visual de dados. Normalmente estes aplicativos possuem características bastante exigentes em termos de critérios estéticos, restrições de desenho e principalmente eficiência, uma vez que exigem tomadas de decisões, na maioria das vezes, em tempo real.

Para se atingir este objetivo, a abordagem para desenho de grafos ortogonais, denominada *topologia-forma-métrica,* foi estudada. Esta abordagem consiste em tratar o desenho em três etapas: *planarização*, *ortogonalização* e *compactação*. Por se tratar de um problema NP-Difícil, cada etapa é resolvida por heurísticas que visam atender a determinados critérios estéticos. Na maioria das vezes, estes critérios estéticos conflitam entre si, o que mostra tratar-se de um problema de otimização. Como são vários os critérios estéticos a serem considerados, pode-se claramente utilizar técnicas de otimização multicritério. Esta abordagem possui os seguintes problemas em aberto: i) não permitir o tratamento de mais de um critério estético por etapa; ii) fixar o embutimento planar que será submetido as próximas etapas. Isto não garante um desenho otimizado ao final do processo; ii) existência de dependências entre as três etapas da abordagem, exigindo que elas sejam executadas em uma ordem pré-definida. Estes problemas serão solucionados neste trabalho.

As principais contribuições do trabalho são: i) a obtenção dos modelos matemáticos que representam diversos critérios estéticos e o tratamento dos mesmos por técnicas de programação linear inteira (PLI) e otimização multicritério na etapa de compactação; ii) o desenvolvimento de três abordagens híbridas utilizando a *topología-forma-métrica* com o algoritmo genético, e iii) o desenvolvimento de uma metodologia unificada que trabalha simultaneamente com os critérios de minimização de cruzamentos, dobras e a soma total das arestas, com o auxílio do algoritmo genético e de operadores de busca local.

A metodologia por PLI gerou resultados melhores que a abordagem do fluxo de custo mínimo em rede quando aplicada considerando apenas um critério estético na compactação. Quando utilizada em um contexto com os vários objetivos modelados, gerou bons resultados respeitando os critérios estéticos e suas restrições. As abordagens híbridas resolveram o problema de fixar o embutimento planar existente na *topologia-forma-métrica* e apresentou melhorias de aproximadamente 50%, quando comparada aos resultados da abordagem clássica, para o pior caso tratado. A abordagem unificada, além de eliminar a interdependência entre as etapas, conforme ocorre na *topologia-forma-métrica*, apresentou bons resultados para a solução de problemas de desenhos ortogonais no *grid*. Além disto, ela garante flexibilidade para o tratamento do número de critérios estéticos e restrições necessárias e independe das características do modelo matemático que os representam, uma vez que trabalha em conjunto com o algoritmo genético.

ABSTRACT

The general goal of this work is the study and development of methodologies for automatic graph drawings in order to assist the solution of important problems related to the quality, readability, reliability and visibility of information provided by applications that use resources related to the visual representation of data. Typically these applications have characteristics very demanding in terms of aesthetic criteria, constraints and mainly efficiency, given that they often require real time decision making.

To achieve this goal, the approach for orthogonal graph drawings called *topology-shape-metric* has been studied. This approach consists in treating the drawing in three steps: planarization, orthogonalization and compaction. Given that each step represents an NP-hard problem, each step is handled by heuristics that aim at resolving certain aesthetic criteria. Most often these aesthetic criteria conflict with each other. Since we have several aesthetic criteria that must be treated, one can clearly deal with them using multicriteria optimization techniques. This approach has some opened problems that will be solved in this work.

This work presents the following contributions: i) mathematical models that represent different aesthetic criteria and its solution by integer linear programming (ILP) and multicriteria optimization techniques in the compaction step are obtained; ii) three hybrid approaches considering the topology-shape-metrics and genetic algorithm are developed and iii) a unified methodology for obtaining orthogonal graph drawings on the grid is developed. This is related to a new methodology, considered unified because it solves, simultaneously, the aesthetic criteria: crossings minimization, bends minimization and the total sum of the edges length minimization with genetic algorithm and local search algorithms.

The methodology for ILP showed better results than the approach of minimum cost flow in the network when applied considering only one aesthetic criterion in compaction step. When used in context with the variety of goals modeled, showed interesting results. The

hybrid approaches solved the problem of fixing the planar embedding in the *topology-shape-metric* and showed improvements of approximately 50% compared to the classical approach results, for the worst case treated. Moreover, it ensures flexibility to address the number of aesthetic criteria and constraints necessary. It is independent of the mathematical model characteristics that represent them, since it works in conjunction with the genetic algorithm.

SUMÁRIO

LISTA DE SIGLAS E ABREVIATURAS

AG	-	Algoritmo Genético (do inglês Genetic Algorithm - GA)
ANEEL	-	Agência Nacional de Energia Elétrica
APG	-	Algoritmo para planarização de Auslander, Parter e Goldstein.
CAD	-	Desenho Auxiliado por Computador.
CEMIG	-	Companhia Energética de Minas Gerais
$COMP_{soma}$	-	Minimização da soma das arestas.
$COMP_{área}$	-	Minimização da área do desenho.
$COMP_{max}$	-	Minimização da maior aresta.
GTAD	-	Graph Toolkit for Algorithms and Drawings.
LEC	-	Algoritmo para planarização de Lempel, Even e Cederbaum.
NP-Difícil	-	Non-Polinomial (difícil).
NSGA	-	Non-Dominated Sorting Genetic Algorithm
NSGAII	-	Algoritmo Genético Multiobjetivo (NSGAII – Fast Non-
	-	Dominated Sorting Genetic Algorithm)
PERT	-	Program Evaluation and Review Technique (diagramas).
P&D	-	Pesquisa e Desenvolvimento
PLI	-	Programação Linear Inteira.
PMO	-	Problema Multiobjetivo
PPGEE	-	Programa de Pós Graduação em Engenharia Elétrica.
SCADA	-	Supervisory Control and Data Acquisition
UFMG	-	Universidade Federal de Minas Gerais.
UML	-	Unified Modeling Language
VLSI	-	Very-Large Scale Integration (criaçao de circuitos integrados).

LISTA DE SÍMBOLOS

A	-	Árvore com raiz.
a(f)	-	O número total de *ângulos-vértices* dentro da face f do embutimento planar Γ.
c	-	Vetor c corresponde aos tamanhos (coordenadas) a serem atribuídos.
C_v (x_v, y_v)	-	Coordenadas x e y dos vétices v
$D(v)$	-	O conjunto de semi-arestas que iniciam no vértice v.
$D(f)$	-	O conjunto de semi-arestas da face f que são percorridos no sentido anti-horário.
D_x	-	Grafos direcionados horizontalmente.
$D(x)$	-	Solução *fuzzy*
D_y	-	Grafos direcionados verticalmente.
$e(f,g)$	-	Arestas entre duas faces (f e g).
$e(u,v)$	-	Aresta que liga vértice u no vértice v.
E_h	-	Conjunto de arestas horizontais.
E_v	-	Conjunto de arestas verticais.
F_i	-	Conjunto de soluções na fronteira i
F_{in}	-	Conjunto de faces internas.
F_{out}	-	Face externa.
$F_i(x)$	-	Conjuntos *Fuzzy*
f_B	-	Número de dobras
f_L	-	Soma total das arestas
f_{SE}	-	Número de sobreposições entre arestas
f_{SEV}	-	Número de sobreposições entre arestas e vértices
f_{SV}	-	Número de sobreposições entre vértices
f_X	-	Número de cruzamentos
G	-	Grafo G.
$G = (V,E)$	-	Grafo G formado pelo conjunto de vértices V e arestas E.
$G' = (V',E')$	-	Subgrafo G' do grafo G.
$G\Gamma = (F,E)$	-	Grafo dual (F faces e E arestas).
H	-	Grafo Gerador.
$\boldsymbol{H}$	-	Representação ortogonal de um grafo G.
$K_{3,3}$	-	Subgrafo de Kuratowski que contém a subdivisão $K_{3,3}$.
K_5	-	Subgrafo de Kuratowski que contém a subdivisão K_5.
$l(s)$, $r(s)$, $b(s)$ e $t(s)$	-	Limites de um sub-segmento (mais à esquerda, mais à direita, mais abaixo e mais acima).
$l_{(u,v)}$	-	Número de dobras para a esquerda ($3\pi/2$).
$\max \mu_D(x)$	-	Grau máximo de pertinência *Fuzzy*
$\mathcal{N}$	-	Rede.

n	-	Número de gerações atual
nMax	-	Máximo de gerações para o algoritmo
N_{hor}	-	Fluxo de rede para segmentos horizontais.
N_{ver}	-	Fluxo de rede para segmentos verticais.
$N(\mu,\sigma)$	-	Distribuição de probabilidade Gaussiana
P	-	População Pai
Q	-	População filha
$r_{(v,u)}$	-	Número de dobras para a direita ($\pi/2$).
S_{h}	-	Segmentos horizontais.
S_{v}	-	Segmentos verticais.
trans(A)	-	Fecho transitivo do conjunto de arcos A.
$\alpha(u,v) \cdot \pi/2$	-	É o ângulo no vértice u formado pelo primeiro segmento da semi-aresta (u,v) e o próximo segmento no sentido anti-horário em torno de u.
$\alpha(u,v)$	-	Representa a medida do ângulo formado pelo vértice u dentro da face f.
$\beta(u,v)$	-	Representa o número de dobras com ângulo igual a $\pi/2$ na face f, ao longo da aresta (u,v) entre as faces f e g.
Γ	-	Embutimento de um grafo planar.
δ_h	-	Conjunto de sub-segmentos horizontais.
δ_v	-	Conjunto de sub-segmentos verticais.
Δx	-	Variação em x
Δy	-	Variação em y
ξ	-	Tamanho do aumento na zona de busca
$\lambda(u,v)$	-	Limite inferior em cada arco (u,v).
μ	-	Média aritmética populacional
$\mu_D(x)$	-	Função de Pertinência de solução *fuzzy*
$\mu(u,v)$	-	Capacidade em cada arco (u,v).
$\sigma = \langle (D_h, D_v) \rangle$	-	Descrição de forma de ***H***.
σ	-	Desvio Padrão
σ^2	-	Variância
$\sigma(v)$	-	Produção / Consumo em cada vértice de saída / entrada.
$\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$	-	Extensão completa de uma descrição de forma σ.
ϕ	-	Fluxo em uma rede $\mathcal{N}$.
$\phi(u,v)$	-	Fluxo no arco (u,v).
$\chi(u,v)$	-	Custo associado a cada arco (u,v).
ψ	-	Desenho ortogonal planar no *grid*.

LISTA DE ALGORITMOS

CAPÍTULO 1

INTRODUÇÃO

A visualização de informações apresentadas através de desenhos de grafos é uma área de pesquisa relativamente nova. Pesquisas nesta área têm como objetivo o estudo de algoritmos e técnicas que possam prover meios de representação visual de dados, normalmente abstratos ou complexos demais, para uma melhor compreensão dos mesmos [1].

O argumento padrão utilizado nesta área é que a exploração do processamento visual pode ajudar as pessoas na compreensão dos dados ou em outras tarefas relacionadas. Isso se deve ao fato que, no cérebro, 70% de todos os receptores e mais de 40% do córtex são destinados à visão. Ou seja, a tentativa de localização de padrões dentro de dados alfanuméricos requer maiores inferências da memória do que a mesma tarefa realizada com auxílio de recursos visuais [2].

A visualização de informações é caracterizada pela necessidade do projetista de criar uma representação gráfica dos dados. Esta representação deve expressar as características mais relevantes dos dados, o que em geral requer a demonstração de relacionamentos. Isto pode ser simples, como em um gráfico de setores para representar a divisão de um mercado entre fornecedores. Outras vezes é bastante complexo como, por exemplo, as ligações entre centenas de páginas web de um determinado site. O desenvolvimento de sistemas de visualização deve considerar a melhor forma de mapear os dados para uma representação que facilite a interpretação por parte do usuário. Além disso, devem ser previstos meios que limitem a quantidade de informações exibidas, porém mantendo os usuários informados sobre o conjunto global dos dados.

A representação dos dados e a forma como eles se relacionam levam ao conceito de grafo, já que um grafo representa dados que possuem relacionamentos entre si [3], [4], [5], [6], [7], [8]. Como exemplos de uso, podem ser citados modelos de sistemas de software, de mapas de sites, e de sistemas de tempo real, entre outros. Este conceito, intuitivamente, leva à noção de um diagrama de nós conectados. Entretanto, há uma infinidade de formas de se representar visualmente um grafo [9 - 19].

Os grafos podem representar modelos físicos como, por exemplo, um circuito elétrico, um circuito integrado ou uma rede de computadores. Suas entidades e relações são designadas, respectivamente, por vértices e arestas, os quais podem possuir atributos adicionais como peso, cor, tamanho entre outros.

Nas aplicações relacionadas com a visualização de informações, a utilidade do desenho de um grafo depende da sua legibilidade, ou seja, da sua capacidade de transportar o significado do diagrama rápida e claramente. Problemas com o mau aproveitamento do espaço e a navegação em grafos com tamanhos grandes, prejudicam a legibilidade.

Os algoritmos para desenho de grafos levam em conta as suas propriedades combinatórias como conectividade, biconectividade, planaridade e etc [3], [4], [5], [13], [14]. Para o desenvolvimento desses algoritmos é importante conhecer se o grafo é direcionado e acíclico, se é uma árvore, ou se é planar [3], [4], [5], e [6]. Alguns algoritmos para desenho de grafos trabalham somente (ou melhor) em grafos que pertençam a uma classe específica. Além disto, o usuário geralmente quer um desenho de um grafo que ilustre as suas propriedades combinatórias.

Os algoritmos para desenho também consideram o conjunto de critérios estéticos que se deseja tratar. Um desenho de um grafo pode ser gerado por diferentes abordagens, as quais estabelecem relações de precedências variadas entre critérios estéticos. Assim, o contexto da aplicação deve sugerir características desejadas para o desenho e o melhor algoritmo a ser utilizado [6].

Na literatura são encontrados diversos algoritmos para o tratamento de desenho de grafos, considerando, por exemplo, a abordagem *hierárquica* que envolve modelos de dependências entre relacionamentos, cujo desenho respeita a hierarquia necessária. A abordagem *dirigida por força* que leva em conta métodos intuitivos para se criar um desenho de linhas retas em um grafo não direcionado. A abordagem *dividir para conquistar*, que consiste em primeiro dividir o grafo em subgrafos, depois os subgrafos são recursivamente desenhados, e, finalmente, o desenho de todo o grafo é obtido pela colagem dos desenhos dos subgrafos [18].

Desenhos ortogonais são compostos somente por arestas na horizontal e na vertical. Eles são utilizados em uma gama de aplicações, como: engenharia de software para representar estruturas modulares de programas, em ferramentas de desenho auxiliado por computador (CAD), em representações de circuitos elétricos, entre outros. Nesses desenhos utiliza-se a convenção de desenho no *grid*, na qual as coordenadas dos vértices e dobras das arestas são números inteiros. Para estes desenhos foi, primeiramente, desenvolvido o algoritmo conhecido como algoritmo de Tamassia [17]. Trata-se do

algoritmo para *ortogonalização* de um grafo planar que minimiza o número de dobras nas arestas do desenho. Tamassia mostrou que encontrar uma representação ortogonal de um embutimento planar fixo, com o número mínimo de dobras, é equivalente a encontrar o fluxo de custo mínimo em rede. Este algoritmo recebe como entrada um embutimento planar cujos vértices devem ter grau máximo igual a 4, e devolve uma representação ortogonal do grafo em tempo $O(V^2 \log V)$, sendo V o número de vértices do grafo [17]. Este algoritmo representou um passo importante no desenvolvimento da área de desenho de grafos ortogonais. Posteriormente, em Tamassia et al [19], foram mostrados os algoritmos desenvolvidos para aplicações baseadas na abordagem chamada *topologia-forma-métrica* [16], [17], [18], e [19], onde são consideradas uma gama maior de critérios estéticos e restrições de desenho. Esta metodologia divide o problema em três etapas, a saber: primeiro cuida da *topologia* do desenho, depois da sua *forma*, que deve ser ortogonal, e, finalmente, a *métrica*, onde as coordenadas dos vértices e dobras são atribuídas e os tamanhos das arestas são calculados. Estes algoritmos garantem resultados muito bons sob o ponto de vista de alguns critérios estéticos [6], [18], porém existem limitações: i) não é possível tratar mais de um critério estético por etapa; ii) é considerado apenas um embutimento planar fixo na etapa de planarização, o que não garante um desenho otimizado ao final do processo; iii) exige que as etapas da abordagem sejam executadas na ordem pré-estabelecida, primeiro a *topologia*, depois a *forma* e finalmente a *métrica*.

O desenvolvimento de novas metodologias para desenhos ortogonais no *grid* é o foco principal deste trabalho. Isto se dá devido a necessidade da concessionária de energia elétrica, CEMIG Distribuição SA, em desenvolver uma ferramenta para o desenho dos diagramas sinóticos das subestações de forma automática. Os diagramas sinóticos, ou simplesmente sinóticos, compreendem a representação das subestações (SE'S) no âmbito do sistema SCADA (do inglês *Supervisory, Control and Data Acquisition*), xOMNI, utilizado nos centros de controle da CEMIG-D. Tais sinóticos são associados às informações do sistema elétrico em tempo real, que relacionam o desenho (sinótico) aos equipamentos fisicamente instalados no campo. Estes desenhos possuem características de desenho ortogonais. Como eles são utilizados como *interface-homem-máquina* para operação do sistema elétrico, eles devem ser mostrados de forma clara e organizada. Devem ser dispostos de forma a caber em uma tela de computador, portanto, devem ser compactos, devem obedecer a uma determinada razão de aspecto (proporção entre altura e largura do desenho), devem estar bem distribuídos e, consequentemente, devem ser desenhos de boa qualidade.

A *Figura 1.1* mostra um exemplo de uma *interface-homem-máquina* do sistema xOMNI. Neste exemplo, o desenho foi obtido manualmente utilizando uma ferramenta para desenho auxiliado por computador (*MicroStation*).

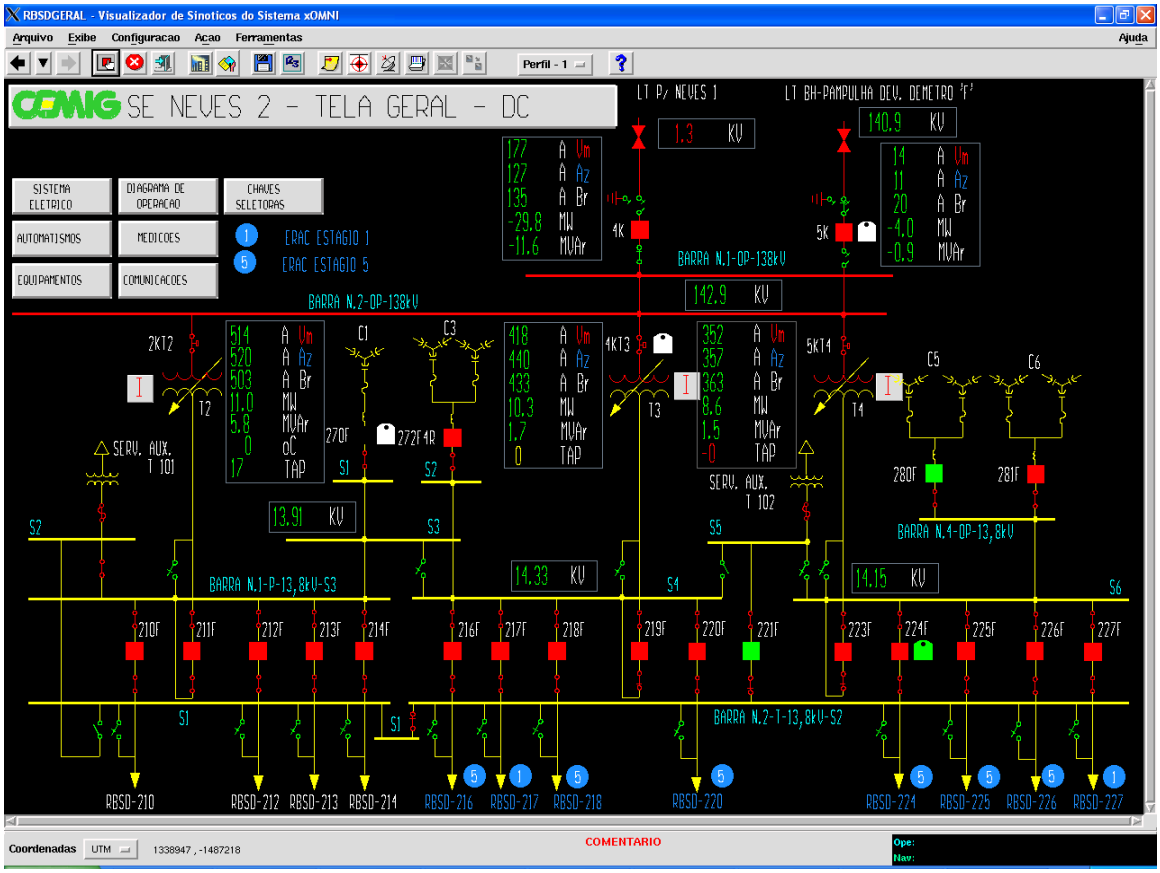


Figura 1.1 - Exemplo de uma Interface Homem Máquina (IHM) do sistema de supervisão e controle da CEMIG-D.

Como se pode observar na *Figura 1.1*, o desenho é basicamente composto por arestas na vertical e na horizontal, o que caracteriza um desenho ortogonal. Ele deve primar por uma boa legibilidade e visibilidade e deve ser compacto o suficiente para caber em uma tela de computador. Além disto, ele deve obedecer a razão de aspecto da tela.

1.1. Justificativas

A principal dificuldade do desenho de grafo é o fato que a obtenção de desenhos sob o ponto de vista de legibilidade e visibilidade está associada a problemas da classe NP-Difícil [13], [14] e [15], ou seja, problemas para os quais não se conhecem algoritmos determinísticos que os resolvam em tempo polinomial. Por esta razão, problemas desta natureza são resolvidos por meio de heurísticas ou meta-heurísticas, o que nem sempre garante a obtenção de resultados ótimos.

Como já dito, a *topologia-forma-métrica*, abordagem mais popular para desenhos ortogonais no *grid*, é dividida em três etapas: 1) a *planarização*, onde o objetivo é minimizar o *número de cruzamentos* entre as arestas do desenho; 2) a *ortogonalização*, na qual cuida-se da *minimização do número de dobras* das arestas; 3) a *compactação*, que visa a minimização da *soma total do comprimento das arestas* do desenho e,

consequentemente, a sua área. As etapas devem ser executadas na ordem explicitada, ou seja, primeiro a *topologia*, depois a *forma* e, por fim, a *métrica*. Esta ordem de importância entre os critérios estéticos se deve ao fato que o número de dobras da *forma* é afetado pela *topologia* e diferentes escolhas de *topologias* podem conduzir a *formas* com diferentes números de dobras. Isto afeta diretamente a legibilidade e visibilidade do desenho final. A abordagem *topologia-forma-métrica* dá prioridade mais alta para o critério estético *minimização de cruzamentos*, que geralmente é considerado o maior responsável pela dificuldade na compreensão do desenho do grafo, do que para a *minimização de dobras*. Como a *topologia* e a *forma* também afetam a *área* do desenho e a *área* do desenho é tratada na etapa de *compactação*, esta deve ser a última etapa da abordagem. Por isto, esta abordagem exige que o desenho seja obtido por esta ordenação de passos, levando a uma interdependência entre as etapas. Além da exigência da ordenação dos passos, os algoritmos existentes para resolver o problema de desenho de grafos na abordagem *topologia-forma-métrica* partem de um embutimento planar fixo obtido na etapa de *planarização* [20]. Fixar o embutimento planar se dá pelo fato de um grafo planar poder ter um número exponencial de embutimentos planares [21]. A *minimização do número de dobras* em todos os possíveis embutimentos de um grafo planar também é um problema NP-Difícil [15].

Além das questões discutidas, o algoritmo mais tradicional para tratar o problema de desenho de grafos na abordagem *topologia-forma-métrica* é baseado no fluxo de custo mínimo em rede e não garante a obtenção do desenho ótimo, considerando os critérios estéticos que são fixos em cada etapa. Não é possível introduzir outros critérios estéticos e nem restrições de desenho nesta abordagem do fluxo de custo mínimo em rede. Alguns algoritmos com base nesta abordagem permitem minimizar a altura e a largura do desenho, mas não garantem que a *soma total do comprimento das arestas* seja mínima ou que a *minimização da área* propicie um desenho com boa legibilidade, e nem mesmo propicie um desenho bem distribuído em uma determinada área mínima [18], [22]. Esta abordagem de desenho de grafos ainda possui muitos problemas em aberto como, por exemplo: necessidade de uma abordagem para tratamento de diversos critérios estéticos simultaneamente; tratamento de critérios estéticos e restrições de desenho desejadas pelos especialistas ou usuários da aplicação; utilização de diversos embutimentos planares distintos na etapa de *planarização* com a finalidade de avaliar se é possível obter desenhos melhores a partir de diferentes embutimentos planares e eliminar ou minimizar a interdependência entre as etapas do desenho, entre outros. Alguns deles, se resolvidos, podem trazer contribuições importantes para a área de desenho de grafos. Solucionar alguns destes problemas é o objetivo principal desta tese, como discutido a seguir.

1.2. Objetivos

Este trabalho tem como objetivo desenvolver técnicas para desenho automático de grafos que tenham por base as seguintes diretrizes:

- Permitir a aplicação de diversas combinações de critérios estéticos e restrições de desenho, tanto os convencionais como os exigidos pelo usuário e resolver o problema multicritério;
- Resolver problemas em aberto na abordagem de desenhos ortogonais no *grid*, *topologia-forma-métrica;*
- Tratar o processo de obtenção de desenho ortogonal no *grid* em apenas uma etapa utilizando algoritmos evolucionários.

Considerando os objetivos definidos, na próxima seção será apresentado um esboço do desenvolvimento do trabalho.

1.3. Esboço do trabalho

A implementação das abordagens sugeridas pelos autores de [22], [23], [24], [25] e [26], que consideram o tratamento do problema de desenho de grafos por técnicas de programação linear inteira (PLI) [27], [28] e [29], foi efetuada neste trabalho. Esta abordagem consiste em transformar as arestas do grafo em segmentos horizontais e verticais, definir uma chamada *descrição de forma* para este conjunto de segmentos e então modelar os critérios estéticos como problema de PLI [30], [31], [32], [33], na etapa de *compactação* da *topologia-forma-métrica*. Foi considerado, inicialmente, apenas o único critério estético abordado pelos autores de [24], [25] e [26] e, posteriormente, estendido para os demais critérios estéticos modelados neste trabalho. Uma vez modelados os demais critérios estéticos com base em programação linear inteira, definidas as funções matemáticas que representam as restrições consideradas, o problema foi então resolvido por técnicas de PLI e técnicas de otimização com mais de um objetivo [34], [35]. A implementação da etapa de *ortogonalização* por PLI utilizada neste trabalho foi a efetuada em [36]. Assim, as técnicas utilizando PLI foram testadas nos novos modelos obtidos e seus resultados são mostrados no Capítulo 3.

Em seguida, resolveu-se o problema existente na *topologia-forma-métrica*, que fixa o embutimento planar a ser submetido às demais etapas da abordagem. Na etapa de *planarização*, um embutimento planar é obtido de acordo com a ordem em que são inseridas as arestas na construção incremental do embutimento. Este embutimento planar, que representa a topologia do desenho, serve de base para as próximas etapas da abordagem. Sendo assim, é importante avaliar se a construção incremental do embutimento

planar oferece ou não um desenho final melhor se a ordem de inserção das arestas for escolhida de maneira diferente da utilizada na abordagem tradicional. Isto leva ao problema de se buscar uma sequência ótima [37], [38] e [39], um problema típico em otimização combinatória. Desta forma, o problema foi modelado como um problema de permutação de inteiros e, consequentemente, foi possível obter um maior número de embutimentos planares, resolvendo, assim, o problema de se fixar o embutimento planar na etapa de *planarização*. Tendo em mãos este conjunto de embutimentos planares, tornou-se necessário a aplicação de técnicas que permitam escolher, dentre eles, o melhor embutimento planar. Técnicas evolucionárias, mais especificamente, o Algoritmo Genético [40], [41], [42], [43], [44], [45], [46] e [47] são indicadas para solução de problemas desta natureza. Assim, o problema de escolha do embutimento planar otimizado é resolvido aplicando operadores de recombinação e de mutação apropriados para representações por permutações e o Algoritmo Genético (AG).

Baseadas nestas idéias, uma abordagem denominada abordagem híbrida I, onde são utilizadas a *topologia-forma-métrica* e o algoritmo genético, foi desenvolvida e é descrita como:

- Dada uma sequência de representação (sequência de inserção das arestas), cria-se o embutimento do grafo, usando o algoritmo de *planarização*;
- Aplica-se a permutação na sequência de representação de forma a gerar um conjunto de embutimentos do grafo;
- Para cada embutimento do grafo, faz-se a *ortogonalização* e a *compactação*;
- Avalia-se a satisfação dos critérios estéticos desejados e escolhe-se o desenho que melhor represente todos os critérios estéticos simultaneamente.

Considerando o conceito do hibridismo entre a *topologia-forma-métrica* e o AG, foi possível desenvolver três algoritmos híbridos que diferem por sofrerem variações no seu processo evolucionário. Seus resultados foram significativos, apresentando melhorias em relação a abordagem clássica de até 50% para o pior caso tratado.

Uma nova abordagem, definida como abordagem unificada, também foi desenvolvida. Esta abordagem utiliza o algoritmo genético para obtenção de desenhos ortogonais no *grid*. Nesta metodologia, visa-se eliminar a interdependência existente entre as etapas da *topologia-forma-métrica*, resolvendo-se o problema de desenho de grafos ortogonais em apenas uma etapa (abordagem II). Neste caso utiliza-se o algoritmo genético definindo o genótipo e os operadores adequados. Obteve-se uma metodologia unificada na área de desenhos ortogonais no *grid*, que, além de proporcionar desenhos mais otimizados, ainda leva a flexibilidade para tratar diversos critérios estéticos e suas restrições. Na próxima seção são sintetizadas as principais contribuições do trabalho.

1.4. Contribuições

O desenvolvimento das metodologias citadas levaram a contribuições originais na área de desenho de grafos que são sintetizadas e enumeradas nesta seção:

1. Desenvolvimento de modelos matemáticos para diversos critérios estéticos de desenho de grafos, na etapa de compactação, permitindo o tratamento dos mesmos com técnicas de Programação Linear Inteira (PLI);

2. Desenvolvimento de três abordagens híbridas entre a *topologia-forma-métrica* e o algoritmo genético. Cada abordagem difere em como o problema é tratado no processo evolucionário do algoritmo genético: i) a primeira considera a soma ponderada dos objetivos como função de aptidão, que permite avaliar a qualidade do indivíduo no algoritmo genético; ii) a segunda considera a metodologia de tomada de decisão multicritério em ambiente *fuzzy* como critério de ordenação das alternativas, durante o processo evolucionário do algoritmo genético, levando em consideração os conceitos de soluções harmoniosas; iii) a terceira utiliza o algoritmo genético puramente multiobjetivo (NSGAII) para obtenção do conjunto solução de Pareto, onde é aplicada a metodologia de tomada de decisão multicritério em ambiente *fuzzy* para escolha da solução mais harmoniosa no conjunto solução de Pareto.

3. Desenvolvimento de uma abordagem unificada com base na utilização de algoritmo genético, para permitir a obtenção do desenho ortogonal no *grid* em apenas uma etapa.

Estes desenvolvimentos foram testados e seus resultados foram publicados e/ou submetidos a congressos e periódicos internacionais conforme mostrado na seção seguinte.

1.5. Publicações e submissões de artigos relacionados aos resultados do trabalho

B. M. M NETA, G. H. D. ARAUJO, F. G. GUIMARÃES and R. C. MESQUITA, A hybrid genetic algorithm for automatic graph drawing based on the topology-shape-metric approach, In **Proceedings of the Genetic and evolutionary Computation Conference (GECCO 2010),** ACM Press, 2010, pp. 743–750.

B. M. M NETA, G. H. D. ARAUJO, F. G. GUIMARÃES, R. C. MESQUITA and P. Ya. EKEL, A Fuzzy Genetic Algorithm for Automatic Orthogonal Graph Drawing, submetido ao **Journal of Applied Soft Computing-ELSEVIER** em setembro de 2010. Aguardando resultado.

Para a abordagem unificada os artigos serão escritos e submetidos futuramente a congressos e/ou periódicos relacionados com a área de estudo.

1.6. Organização do trabalho

Este trabalho foi organizado em 5 capítulos, os quais foram divididos da seguinte forma:

Capítulo 1 – É feita uma introdução ao trabalho.

Capítulo 2 – Alguns conceitos preliminares sobre desenho de grafos são apresentados e a abordagem *topologia forma-métrica* é detalhada.

Capítulo 3 – São apresentadas as novas metodologias por PLI, as novas metodologias híbridas desenvolvidas e os resultados obtidos.

Capítulo 4 – É apresentada a nova metodologia unificada desenvolvida para desenhos de grafos ortogonais no *grid*, bem como os resultados obtidos.

Capítulo 5 – São apresentadas as conclusões e as proposições para trabalhos futuros.

CAPÍTULO 2

DESENHO AUTOMÁTICO DE GRAFOS

O campo de estudos relacionado com desenho automático de grafos provê algoritmos para a construção de representações geométricas embasadas em estruturas representadas por grafos, em que os vértices representam os elementos e as arestas representam o relacionamento entre eles. Aplicações interessantes e munidas de grandes desafios abundam em diversas áreas das ciências [48], [49], como, por exemplo: engenharia de software para representar estruturas modulares de programas, a hierarquia de classes e de objetos (principalmente nas linguagens visuais e nas ferramentas de desenvolvimento de sistemas) [17], [18], [19]; em aplicações de CAD para facilitar a análise e a manipulação de dados; em sistemas de telefonia para ilustrar uma rede de chamadas telefônicas em uma determinada região ou país; em ambientes de comunidade virtual para apresentar redes sociais; na automação do sistema elétrico representando os diagramas unifilares de subestações e alimentadores; em desenhos de circuitos integrados (VLSI) [50], [51]; entre outras. Estes trabalhos são desenvolvidos principalmente em resposta às necessidades de desenvolvimento de técnicas para visualização de dados, interação entre sistemas de computadores e interação entre homem-máquina. Atualmente, basta abrir um livro de ciências ou uma revista científica para perceber que as figuras ou diagramas ilustrativos têm um bom e eficiente trabalho relacionado a desenho de grafos.

Alguns conceitos sobre grafos, desenho de grafos, suas características e as metodologias existentes serão discutidas neste capítulo. A metodologia para desenho de grafos ortogonais, *topologia-forma-métrica,* será apresentada e melhor detalhada porque será utilizada como base para algumas das novas metotologias desenvolvidas neste trabalho. As técnicas para resolução de problemas de desenho de grafos com base em programação linear inteira (PLI) são apresentadas também neste capítulo.

2.1. Grafos

Nesta seção serão apresentadas algumas definições básicas relacionadas com a Teoria dos Grafos as quais podem ser vistas com mais profundidade em [3], [4], [5], [8], [9], [10], [11].

Um *grafo G* (não–direcionado, veja *Figura 2.1* para ilustração) é um par (*V,E*), onde *V* é um conjunto finito de pontos denominados *vértices* e *E* é um conjunto finito de *arestas* interligando vértices. Uma aresta $e \in E$ é um par não ordenado (*x,y*) de vértices distintos de *V* . A aresta *e* pode ser escrita como (*x,y*) ou (*y,x*) e indica que *x* e *y* são os extremos de *e*. Neste caso, diz-se que *x* e *y* são *adjacentes* e que a aresta (*x,y*) é *incidente* em *x* e em *y*. O *grau* de um vértice $x \in V$ é definido como sendo o número de arestas de *G* incidentes a *x*. Um grafo pode ter arestas múltiplas, ou seja, dois vértices adjacentes podem ter mais que uma aresta incidente a eles. Quando uma aresta tem seus dois extremos em um mesmo vértice, denomina-se laço, loop ou *self-loop*. A *Figura 2.1* ilustra os conceitos de grafo não-direcionado, arestas múltiplas (arestas 2,3 e 4) e self-loop que é representado pela aresta 6. Duas arestas são ditas adjacentes se elas partilham (chegada ou saída) um mesmo vértice. Dois vértices são ditos adjacentes se eles partilham uma mesma aresta.

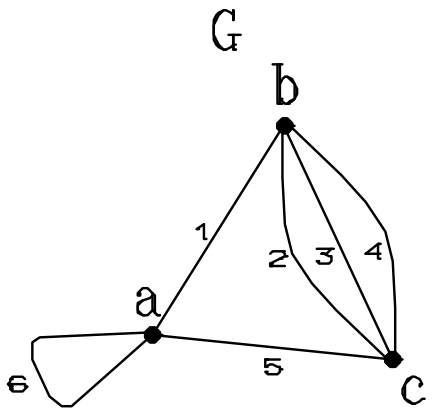


Figura 2.1 - Exemplo de um grafo não direcionado, com arestas múltiplas (2, 3 e ,4) e self-loop (6).

Um grafo *G'* = (*V',E'*) é um *subgrafo* de *G* = (*V,E*), se $V' \subseteq V$ e $E' \subseteq E$. Se *G'* contém todas as arestas de *G* cujos extremos são vértices de *V'*, então *G'* é chamado *subgrafo induzido* de *G*.

Um grafo *H* é dito *gerador* de um grafo *G*, se *H* é um subgrafo de *G*, e quaisquer dois vértices *u* e *v* são adjacentes em *H* se e somente se forem adjacentes em *G*.

Dois grafos são *isomorfos* se há uma correspondência entre seus conjuntos de vértices que preserve a adjacência. Assim, *G* = (*V,E*) é isomorfo a *G'* = (*V',E'*) se existe uma função bijetora $\phi : V \rightarrow V'$, tal que $(x,y) \in E$ implica que $\phi(x)\phi(y) \in E'$ e vice-versa. Naturalmente, grafos isormofos têm o mesmo número de vértices e de arestas.

De forma semelhante à definição de grafo não-direcionado, um *grafo direcionado G* (veja *Figura 2.2* para ilustração) é um par (*V,E*), onde *V* é um conjunto finito de vértices e *E* é um conjunto de pares **ordenados** de vértices. Num grafo direcionado, uma aresta *e* = (*x, y*)

$\in E$ é dita *dissidente* em x e *incidente* em y, respectivamente. O vértice y é dito *adjacente* a x.

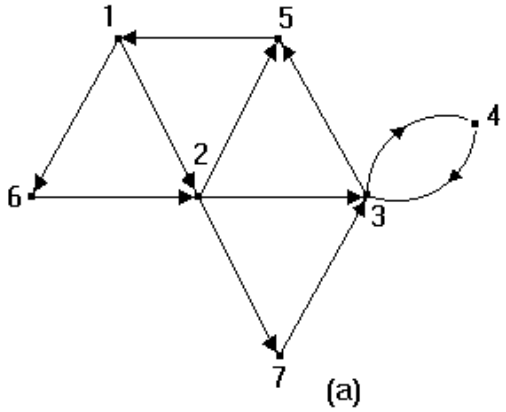


Figura 2.2 - Exemplo de um grafo direcionado.

O *grau de entrada* de um vértice x em um grafo direcionado G é definido como sendo o número de arestas que incidem em x. O *grau de saída* deste vértice é dado pelo número de arestas dissidentes de x. O *grau* de x é definido como a soma entre o grau de entrada e o grau de saída de x.

Um *caminho*, de um vértice x para um vértice y, em um grafo G consiste de uma sequência $S_{x,y} = (x = v_0, v_0v_1, v_1, v_1v_2, \ldots, v_{k-1}v_k, v_k = y)$ em que v_iv_{i+1} são arestas de G (para $i = 0, 1, \ldots, k - 1$) e $v_0, \ldots, v_k$ são vértices de G. Sem perda de generalidade, pode-se omitir os vértices da sequência. Deste modo, um *caminho* de um vértice x para um vértice y é uma sequência de arestas $S_{x,y} = (v_0v_1, v_1v_2, \ldots, v_{k-1}v_k)$, em que $v_0 = x$, $v_k = y$, e v_iv_{i+1} são arestas de G para $i = 0, 1, \ldots, k - 1$. Quando um caminho tem ambos os seus extremos iguais ($v_0 = v_k$), ele é chamado de *caminho fechado ou ciclo ou circuito*. Quando em um caminho cada vértice aparece apenas uma vez, ele é chamado de *caminho ou circuito simples*. O *comprimento* de um caminho $S_{x,y}$ é representado por $|S_{x,y}|$ e é definido como sendo o número de arestas da seqüência. À *distância* entre dois vértices x e y no grafo (também conhecida como *distância teórica*) é dada pelo comprimento do menor caminho de x para y.

Um grafo G é *conexo* se existe um caminho $S_{x,y}$ para todo par de vértices distintos x e y de G. O *centro teórico* de um grafo não-direcionado G é representado por um vértice x cuja soma das distâncias de x a todos os demais vértices de G é mínima.

Para melhor entendimento, será feita uma breve revisão de algumas definições relacionadas com conectividade em grafos. Um *k-conjunto* separável de um grafo G, é um conjunto de k vértices, os quais removidos, desconecta ou diminui o número de componentes conectados de G. As separações em *1-conjunto* são chamadas vértices de

corte (*cut-vértices*) (veja *Figura 2.3*) e em *2-conjunto* são chamadas pares de separação (*split-pairs*) (veja *Figura 2.4*). Um grafo conectado é dito *biconectado* se ele não possui nenhum vértice de corte. Os blocos de um grafo *biconectado* (também chamado de componentes conectadas) são subgrafos biconectados maximal. Um grafo é triconectado se ele é biconectado e não possui pares de separação.

Um *caminho direcionado* de um vértice *x* para um vértice *y* é uma seqüência de arestas $S_{x,y} = ((v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k))$, em que (v_i, v_{i+1}) são arestas orientadas de *G*, para $i = 0, 1 \ldots k - 1$. Quando um caminho direcionado tem ambos os seus extremos iguais ($v_0 = v_k$), ele é chamado *caminho direcionado fechado* ou *ciclo direcionado*.

Um grafo direcionado é dito *acíclico* (também conhecido por *DAG*, do inglês "*Directed Acyclic Graph*") se não possui caminhos direcionados fechados. Se existe um caminho direcionado de um vértice *x* para um vértice *y* em um grafo direcionado *G*, então se diz que *y* é *alcançado* por *x*. Defini-se o *conjunto alcançável* do vértice *x*, notação R_x, como sendo o conjunto de todos os vértices alcançáveis por *x*.

Um grafo G_s não-direcionado é *subjacente* a um grafo direcionado *G* se G_s é obtido a partir de *G* removendo-se somente a orientação de suas arestas.

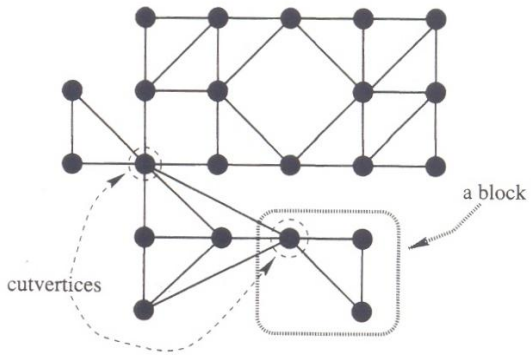


Figura 2.3 - Exemplo de vértices de corte e blocos [15].

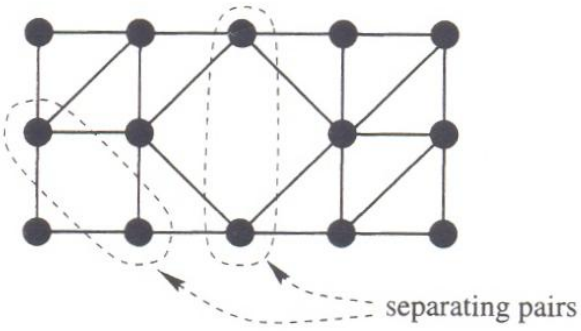


Figura 2.4 - Exemplo de par de separação [15].

Um grafo direcionado é *fracamente conexo* se seu grafo subjacente for conexo. Os grafos direcionados fracamente conexos serão referenciados, neste trabalho, apenas por "grafos conexos", suprimindo-se o termo "fracamente".
Para um grafo $G = (V,E)$, direcionado ou não–direcionado, denota-se por $|V|$ e $|E|$ os tamanhos dos conjuntos V e E, respectivamente. O *tamanho* de um grafo G, notação $|G|$, consiste da soma $|V| + |E|$.

Uma *árvore* é um grafo $A = (V,E)$ conexo onde $|E| = |V| - 1$. Todo vértice de uma árvore, com grau 1 (um), é chamado de *folha*. A *Figura 2.5* mostra o exemplo de uma árvore.

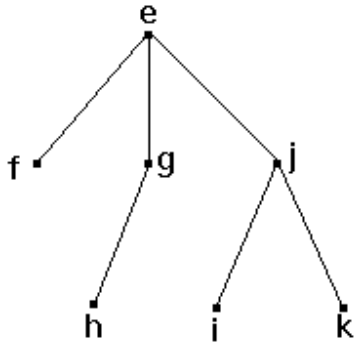


Figura 2.5 - Exemplo de uma árvore.

Uma *árvore com raiz* é uma árvore $A = (V,E)$ que possui um vértice $r \in V$, chamado de *raiz*, e uma ordem hierárquica dos seus vértices que inicia em r e termina nas folhas. A profundidade de um vértice $v \in V$ é dada pela distância de v a r. A altura de A é a maior profundidade de seus vértices. A hierarquia é definida de acordo com a profundidade de cada vértice.

Os *filhos* de um vértice $v \in V$ são os vértices adjacentes a v que possuem profundidade maior do que ele; nesse caso, v é chamado de *pai* desses vértices. A árvore A é dita *binária* se a raiz tem grau máximo 2 (dois) e todos os demais vértices possuem grau menor ou igual a três.

2.2. Desenho de grafos planares

Um *grafo planar* é aquele que pode ser embutido no plano de maneira que suas arestas somente se interceptem geometricamente em seus vértices extremos.

Um grafo é dito *planar maximal* se é um grafo planar e não é possível adicionar mais arestas sem violar a sua planaridade.

Um desenho *planar* particiona o plano em regiões conectadas, chamadas *faces*. A face ilimitada é chamada de face externa, enquanto que as outras são chamadas de faces internas.

Existe uma fórmula simples relacionada ao número de vértices, arestas e faces em um grafo planar conectado, a fórmula de Euler. Euler a estabeleceu para aqueles grafos planares definidos pelos vértices e arestas de um poliedro. A formula de Euler implica que um grafo planar com n vértices, m arestas e f faces (inclusive a face externa) obedece a relação $n - m + f = 2$. Além disto ele estabeleceu que grafos planares são *esparsos*, pois um grafo planar simples com n vértices tem no máximo $3n - 6$ arestas.

2.2.1. Embutimento planar

Um *embutimento planar* de um grafo é uma classe de equivalência de desenhos planares descrita pela ordem circular, em sentido horário, dos vizinhos de cada vértice [17] e [18]. A *Figura 2.6* ilustra quatro desenhos planares de um mesmo grafo, sendo que (a), (b) e (c) têm o mesmo embutimento planar e (d) tem um embutimento planar diferente dos demais. Um grafo embutido é um grafo com um embutimento específico e um grafo planar pode ter um número exponencial de embutimentos [21].

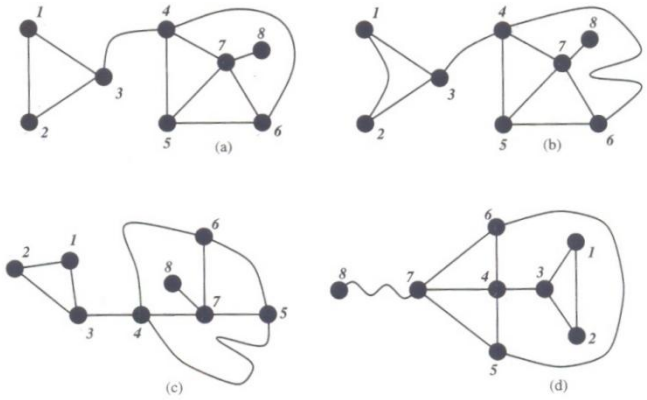


Figura 2.6 - Quatro desenhos planares do mesmo grafo [18].

Algumas técnicas para desenho de grafos planares são baseadas na construção incremental do desenho e utilizam, às vezes, o dual do grafo planar para minimizar o número de cruzamentos entre as arestas do grafo.

2.2.2. Grafo dual

.

Seja G um grafo planar e Γ o embutimento planar de G. O grafo dual $G_\Gamma = (F,E)$ de G em relação a Γ é definido como [5], [6], [18]:

- O conjunto de vértices *F* corresponde ao conjunto de faces em Γ.
- Existe uma correspondência *um-para-um* entre as arestas de Γ e as arestas de seu dual G_Γ. Seja *e* uma aresta de Γ e *e'* a aresta correspondente em G_Γ. Assim, *e'* conecta os vértices de G_Γ que correspondem às faces de Γ que têm *e* como seu limite.

Em palavras mais simples, o grafo dual G_Γ de um embutimento Γ de um grafo planar *G* tem um vértice para cada face de *G* e uma aresta (*f*,*g*) entre duas faces, *f* e *g*, para cada aresta que é compartilhada por *f* e *g*. Um grafo planar e seu grafo dual são mostrados na *Figura 2.7*. O grafo dual G_Γ captura a informação combinatória no embutimento. Se dois desenhos têm o mesmo embutimento, então eles têm o mesmo grafo dual. Um grafo dual pode ter arestas múltiplas e self-loops.

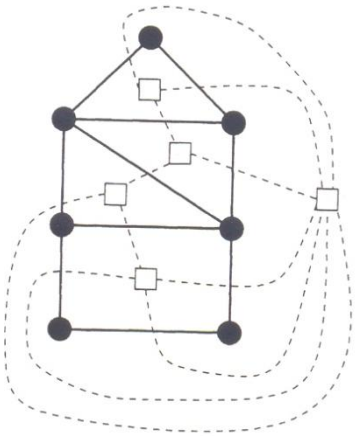


Figura 2.7 - Um grafo dual de um embutimento de um grafo planar (quadrados para os vértices e linhas tracejadas para as arestas) [18].

2.2.3. Paradigmas em desenho de grafos

A entrada para um algoritmo de desenho de grafos é o grafo *G* a ser desenhado. Para desenhar *G* é importante levar em conta algumas de suas propriedades combinatória. Por exemplo, pode-se saber que *G* é direcionado e acíclico, ou que é uma árvore, ou que é planar [6]. Geralmente a classe de grafos, a qual *G* pertence, é conhecida. Este conhecimento é importante por pelo menos duas razões [18]:

- Alguns algoritmos para desenho de grafos trabalham somente (ou melhor) em grafos que pertençam a uma classe específica.
- O usuário geralmente quer um desenho de *G* que ilustre as propriedades combinatória de *G*. 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.

A classe de um grafo é um parâmetro de entrada essencial para uma metodologia de desenho de grafos.

A necessidade de outros tipos de parâmetros cresce da observação de que "o melhor" desenho de um grafo pode até não existir. A percepção humana para um mesmo desenho varia de pessoa para pessoa e diferentes tipos de aplicações requerem diferentes tipos de desenhos. Um outro parâmetro essencial para a metodologia de desenho de grafos é o ambiente particular onde o desenho será utilizado.
Os requisitos do domínio de uma aplicação podem ser modelados em termos de convenção de desenho, um conjunto de estéticas e um conjunto de restrições de desenho. Estes são os parâmetros fundamentais para metodologia de desenho de grafos.

2.2.4. Convenção de desenho

Uma convenção de desenho é uma regra básica que um desenho deve satisfazer para ser admissível. Por exemplo, pode-se 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 uma aplicação real pode ser muito complexa e pode envolver muitos detalhes de um desenho [18]. Algumas das convenções de desenho utilizadas são:

- **Desenho de polilinhas:** Cada aresta é desenhada como uma "cadeia" poligonal.
- **Desenho em linhas retas:** Cada aresta é desenhada como um segmento de retas.
- **Desenho ortogonal:** Cada aresta é desenhada como uma "cadeia" poligonal de segmentos horizontais e verticais alternadamente.
- **Desenhos no "*grid*":** Vértices, cruzamentos e arestas com dobras possuem coordenadas inteiras.
- **Desenho planar:** Não há cruzamentos entre as arestas.
- **Desenho direcionado para cima (*upward*) ou para baixo (*downward*):** Um desenho de um grafo direcionado em linhas poligonais é chamado de *upward*, quando todas as suas arestas possuem pelo menos uma componente da linha poligonal voltada para cima e nenhuma componente voltada para baixo. O conceito de *upward* pode ser utilizado para referenciar um desenho de um grafo direcionado em que todas as arestas estão voltadas para um único sentido, seja esse para baixo (*downward*) ou para cima (*upward*).

Linhas retas e desenhos ortogonais são casos especiais de desenho de polilinhas. Desenhos de polilinhas possuem grande flexibilidade, desde que eles possam aproximar desenhos que possuem arestas com dobras. Porém, arestas com mais que duas ou três dobras podem ser difíceis de se observar visualmente. Desenhos ortogonais 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.

Na maioria das vezes, entre todos os possíveis desenhos, o interesse está apenas naqueles que são "legíveis", ou seja, que ajudam na visualização dos vértices e arestas e, portanto, na compreensão do grafo.

2.2.5. Critérios Estéticos

Propriedades estéticas gráficas específicas de um desenho são as que se desejam aplicar o máximo quanto seja possível para alcançar maior legibilidade no desenho [6], [18], [52], [53], [54]:

- **Cruzamentos:** É a *minimização do número total de cruzamentos entre arestas*. O ideal seria ter desenhos planares, mas nem todo grafo é planar.
- **Área:** É a *minimização da área* de um desenho. A habilidade para construir desenhos com áreas pequenas é essencial no que se refere à visualização em aplicações práticas, nas quais a economia de espaço em tela é de extrema importância, principalmente quando a convenção de desenho no *grid* é adotada. Além disto, aplicações práticas podem conter grafos relativamente grandes, os quais necessitam ser inteiramente exibidos na tela do computador. Desta maneira, a *minimização de área* deve ser levada em conta sempre que possível. A área de um desenho pode ser formalmente definida de diferentes maneiras. Por exemplo, pode-se defini-la como a área da capa convexa de um polígono (Fecho Convexo), ou a área do menor retângulo que engloba todo o desenho.
- **Comprimento total das arestas:** É a *minimização da soma dos comprimentos das arestas.* Esta estética é significante assim como no caso da área. Desta maneira, deve-se tentar minimizar a soma total dos comprimentos das arestas quando a convenção de *grid*, por exemplo, for adotada. Neste caso, o desenho não pode ser reduzido ilimitadamente.
- **Máximo comprimento da aresta:** É a *minimização do comprimento máximo de uma aresta*. Se o desenho adotar, por exemplo, a convenção de desenho no *grid*, este critério estético torna-se de suma importância, pois caso este desenho possua uma aresta muito grande, ele não poderá ser exibido inteiramente na tela do computador.

- **Comprimento uniforme de arestas:** É a *minimização da variância dos tamanhos das arestas*.
- **Número total de dobras:** É a *minimização do número total de dobras* ao longo das arestas. Esse critério estético é especialmente importante para desenhos ortogonais, enquanto é trivialmente satisfeito por desenho de linhas retas.
- **Dobras uniformes:** É a *minimização da variância do número de dobras nas arestas*.
- **Razão de Aspecto:** É a *minimização da "razão de aspecto"* do desenho, isto é, da relação entre o comprimento do maior lado e o comprimento do menor lado do menor retângulo que contenha o desenho do grafo. Um desenho com "razão de aspecto" alta pode não ser convenientemente colocado em uma interface de uma estação de trabalho, mesmo se tiver área modesta.
- **Simetria:** Mostra as simetrias do grafo no desenho. Essa estética pode ser importante em alguns contextos. Existem modelos matemáticos que definem precisamente tais simetrias e seus respectivos desenhos [6], [18], [26], [55].

Pode-se dizer que a *qualidade* de um desenho está associada ao grau de satisfação de um conjunto de critérios estéticos. Dependendo do quanto um desenho atende a certo critério, ele pode ser considerado como de boa ou de má qualidade. A busca por desenhos de boa qualidade comumente envolve a satisfação de vários critérios estéticos ao mesmo tempo e pode ser tratada como um problema de *otimização multiobjetivo* [34], [35].

Além de critérios estéticos, pode-se definir também "restrições de desenho" para orientar a elaboração dos desenhos [6], [18]. As restrições de desenho estabelecem regras que devem ser cumpridas necessariamente. São exemplos de restrições de desenhos exigir que um conjunto de vértices seja desenhado em uma determinada posição ou seguindo um padrão gráfico diferente dos demais. Enquanto uma restrição de desenho nunca deve ser desrespeitada, um critério estético pode ser atenuado com um certo grau de tolerância [18].

A busca por desenhos de boa qualidade incorre em muitas dificuldades. A principal delas segue do fato de que a obtenção de bons desenhos frequentemente está associada a problemas da classe *NP-Difícil* [13], [14].

2.2.6. Restrições de desenho

Enquanto convenções de desenho e estética são regras e critérios gerais que se referem ao desenho do grafo como um todo, restrições de desenho se referem a subgrafos específicos e subdesenhos. Por exemplo, pode ser necessário desenhar um diagrama PERT tal que as arestas representem as atividades de um dado caminho crítico alinhado, ou então, desenhar

um diagrama de fluxo de dados tal que os vértices que representam as interfaces sejam colocados no limite exterior do desenho [6], [18], [19], [56].

Restrições de desenho comumente utilizadas em aplicações relacionadas com visualização incluem:

- **Centro:** Colocar um dado vértice próximo ao centro de um desenho.
- **Externo:** Colocar um dado vértice no limite externo do desenho.
- **Cluster:** Colocar um subconjunto de vértices bem próximos.
- **Sequência (esquerda/direita e cima/baixo):** Desenhar um dado caminho horizontalmente alinhado da esquerda para direita (verticalmente alinhado de cima para baixo).
- **Forma:** Desenhar um dado subgrafo com a forma predefinida.

2.2.7. Precedência entre estéticas

A maioria das metodologias para desenho de grafos é baseada nas seguintes observações [18]:

- Estéticas quase sempre conflitam entre si. Assim, escolhas são inevitáveis.
- Mesmo se as estéticas adotadas não conflitarem, quase sempre é difícil lidar com todas ao mesmo tempo.

Por exemplo, suponha que a convenção de desenho de grafos ortogonal seja adotada. Para o grafo da *Figura 2.8* não existe desenho ortogonal no *grid* que atenda simultaneamente os critérios de minimizar o número de cruzamentos e o de número de dobras de forma ótima.

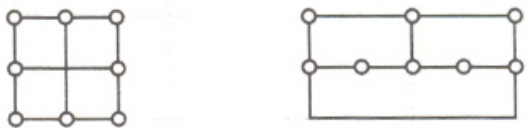


Figura 2.8 - Dois desenhos ortogonais no grid do mesmo grafo: O da esquerda possui o número mínimo de dobras; O da direita possui o número mínimo de cruzamentos [18].

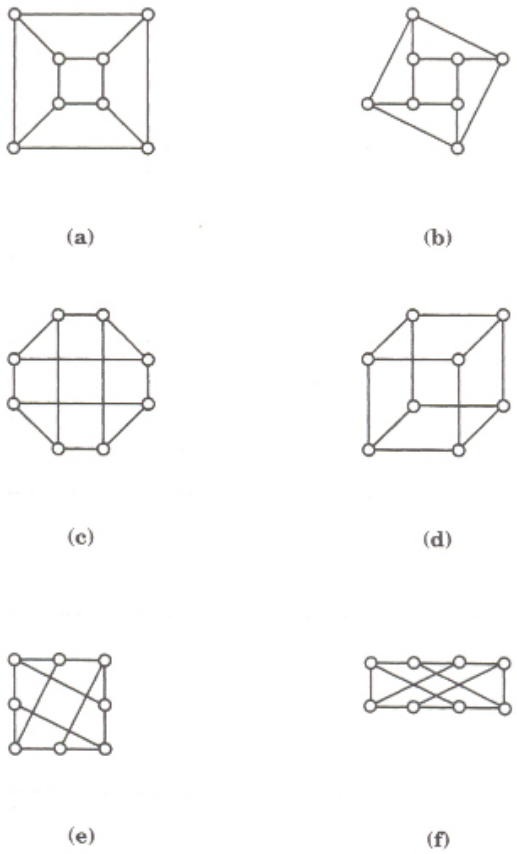


Figura 2.9 - Seis desenhos de um grafo cubo por linhas retas no grid [18].

A *Figura 2.9* mostra seis desenhos no *grid* em linhas retas, do grafo de um cubo. Os desenhos (a) e (b) são planares, enquanto os demais são não planares. Dependendo da definição específica da área adotada (área da capa convexa ou área da capa do retângulo), a área do desenho (a) é maior ou igual à área do desenho (b). O desenho (a) é mais simétrico que o desenho (b). Entre os desenhos não planares (c) a (f), somente os desenhos (c) e (d) são embutíveis no *grid*, visto que os cruzamentos nos desenhos (e) e (f) não possuem coordenadas inteiras. O desenho (f) minimiza a área da capa convexa sobre todos os desenhos de linhas retas com vértices colocados em coordenadas inteiras. Porém, este desenho é esteticamente ruim. Os desenhos (c) e (e) satisfazem ao ciclo Hamiltoniano (ciclo simples que permita passar por todos os vértices sem repetir nenhum deles). Talvez o desenho mais satisfatório seja (d) porque ele mostra a projeção de um cubo tridimensional.

De acordo com a discussão anterior, percebe-se que a maioria das metodologias estabelece uma relação de precedência entre estéticas. Tais relações de precedência são satisfatórias para determinadas aplicações e menos ou não satisfatórias para outras. As abordagens apresentadas na literatura usualmente dividem o desenho dos grafos em processos de

sequência algorítmica dos passos, cada um projetado para satisfazer a certas subclasses de estéticas. Daqui para frente, serão descritas as mais populares destas metodologias. Aquela que será utilizada como base para o desenvolvimento de algumas das novas metodologias apresentadas neste trabalho, a *topologia-forma-métrica*, será detalhada.

2.3. Metodologias para desenho de grafos

Nesta seção serão apresentadas, de maneira simplificada, as metodologias existentes na literatura sobre desenho de grafos. A metodologia *topologia-forma-métrica*, que será utilizada neste trabalho, será detalhada na seção seguinte.

2.3.1. Metodologia hierárquica

Dígrafos são largamente utilizados em aplicações que envolvem modelos de dependência entre relacionamentos. Como exemplos, podem ser citados: Diagramas PERT, hierarquias é-um e grafos de chamada a subrotinas. Dígrafos acíclicos são usualmente representados com uma convenção de desenho de polilinha ascendente ou descendente. A abordagem hierárquica proposta originalmente é intuitiva [54], [57], [58]. Ela é também tratada em passos, como ilustrado na *Figura 2.10* e descrita a seguir:

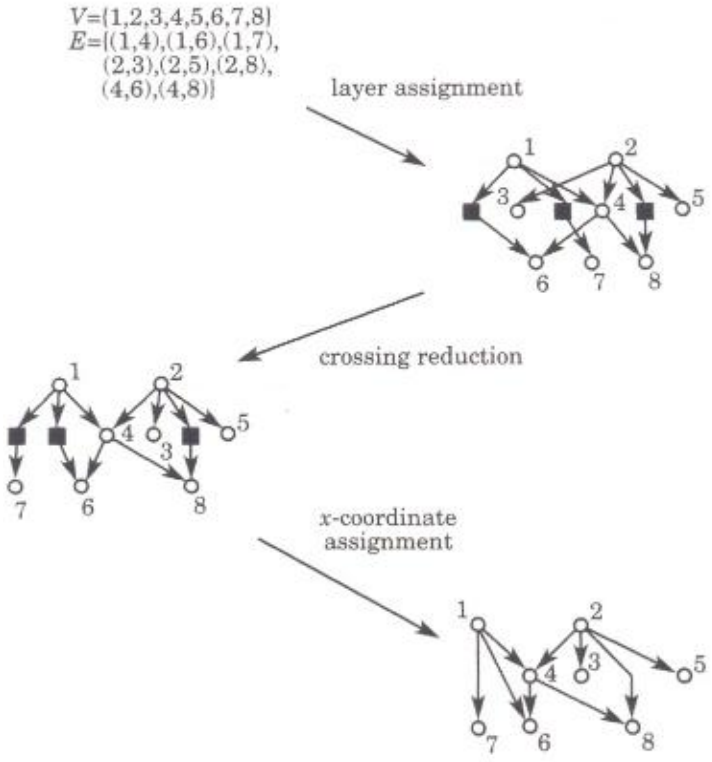


Figura 2.10 - Abordagem hierárquica para dígrafos em geral [18].

1. **Distribuição em camada:** Este passo recebe um dígrafo acíclico como entrada e os seus vértices 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 forma controlada, para 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, pode-se definir que se todas as arestas que estiverem em mais de duas camadas possuírem vértices fictícios de maneira que $i = j + 1$, tem-se 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 as menores possíveis) 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 alto, pois isto afeta a legibilidade do desenho final.
2. **Redução de cruzamentos:** Este passo recebe um "*dígrafo próprio em camadas*" 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 é escolhida de maneira que o número de cruzamentos seja mantido o menor possível. De fato, o problema de *minimização de cruzamentos entre arestas* no dígrafo em camada é NP-Completo, mesmo se ele contiver apenas duas camadas. Assim, uma variedade de heurísticas têm sido utilizadas [54], [57], [58] para *minimização de cruzamentos entre arestas* em dígrafos em camadas.
3. **Atribuição da coordenada x:** 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 é obtido pela representação de cada aresta como um segmento de reta. Então, os vértices fictícios são removidos. Desta maneira, arestas longas podem ser representadas por linhas poligonais.

Algumas estéticas podem ser levadas em conta durante o passo de atribuições das coordenadas *x*. Por exemplo, o vértice fictício introduzido ao longo das arestas no passo de substituição poderá ser alinhado para reduzir o número de dobras no desenho final, ou os vértices poderão ser posicionados horizontalmente para enfatizar simetrias do dígrafo. Os vértices também poderão ser empacotados para reduzir a área do desenho.

2.3.2. Metodologia da visibilidade

A abordagem de *visibilidade* é uma metodologia geral proposta em [59] para desenho de grafos com convenção de desenho por polilinhas. Consiste dos seguintes passos [14], [18] (ver *Figura 2.11*):

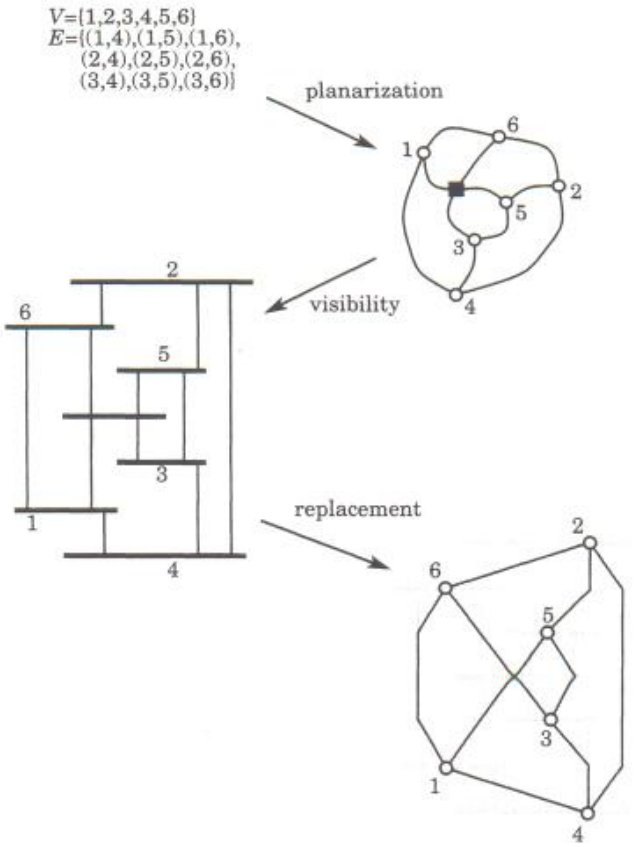


Figura 2.11 - Abordagem de visibilidade [18].

1. O passo de *planarização* é semelhante ao da *topologia-forma-métrica*.
2. **Visibilidade:** neste passo se constrói uma representação de visibilidade do grafo. Na representação de visibilidades, cada vértice é mapeado para um segmento horizontal e cada aresta para um segmento vertical. O segmento vertical, representando a aresta (*u*,*v*), tem seu ponto final no segmento horizontal representando os vértices *u* e *v*. Este não faz interseção com nenhum outro segmento horizontal. Grosseiramente falando, a representação de visibilidade pode ser considerada como um esqueleto ou um esboço de um desenho final.
3. **Substituição:** Neste passo, se constrói o desenho final de polilinhas através da substituição dos segmentos horizontais e verticais na representação de visibilidade como segue. Cada segmento horizontal é substituído por um ponto representando o vértice correspondente e cada segmento vertical é

substituído por uma linha poligonal representando a aresta correspondente, seguindo o segmento vertical original. Existem algumas estratégias de substituição que permitem a construção de um desenho de polilinhas planar a partir de uma representação de visibilidade. A abordagem de visibilidade inicialmente é parecida com a *topologia-forma-métrica* e, por outro lado, o passo de visibilidade é similar ao passo de atribuição em camadas da abordagem *hierárquica*. Neste sentido, a abordagem de *visibilidade* pode ser considerada um ponto intermediário entre as duas abordagens.

O passo de *planarização* reduz o número de cruzamentos, que é um critério estético primário desta abordagem. No passo de visibilidade, é desejável minimizar a área da representação de visibilidade. No passo da substituição, algumas estéticas podem ser levadas em conta, pois existem estratégias que tentam minimizar o número de dobras, estratégias que enfatizam simetrias do grafo e estratégias que balanceiam a distribuição dos vértices no desenho.

2.3.3. Metodologia do aumento

A abordagem do aumento é uma metodologia geral proposta para desenho de grafos na convenção do desenho de polilinhas. A ideia básica é adicionar arestas e/ou vértices ao grafo para se obter um novo grafo com uma estrutura mais forte, e consequentemente, com propriedades mais adequadas para obtenção de desenhos melhores. O método consiste dos seguintes passos [18] (ver *Figura 2.12*):

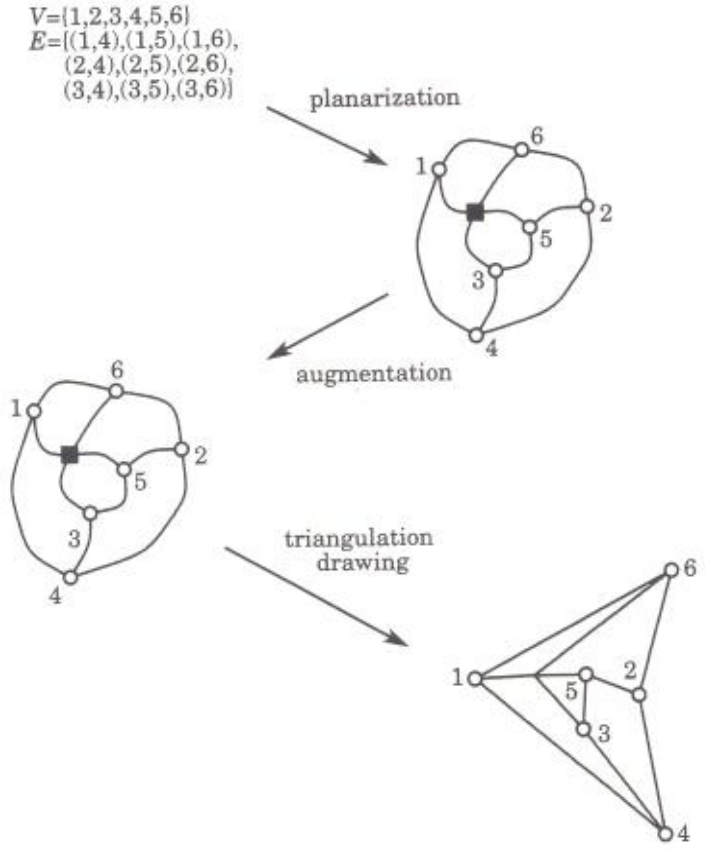


Figura 2.12 - Abordagem do aumento [18].

1. O passo de *planarização* é o mesmo utilizado na abordagem *topologia-forma-métrica*.
2. O passo do aumento adiciona um conjunto satisfatório de arestas (algumas vezes vértices) em um embutimento planar construído no passo anterior, para obter um grafo maximal planar, que é um grafo planar em que as faces têm três arestas. Uma vez que a qualidade do desenho de um grafo maximal planar, em termos de requisitos de área e resolução angular, é usualmente afetado pelo grau dos vértices, pode-se aplicar as técnicas de aumento que **tentam** manter o grau dos vértices tão pequeno quanto possível.
3. **Triangulação do desenho:** Passo que constrói o desenho final pela representação de cada face como triângulos se possível.

Durante os passos do aumento e da triangulação do desenho, estratégias podem ser usadas para minimizar a área, maximizar a resolução angular e distribuir os vértices uniformemente. A abordagem do aumento é menos satisfatória que outras para suportar restrições de desenho. Nem sempre é possível atingir um desenho final com todas as faces triangulares.

2.3.4. Metodologia dirigida por força

Os algoritmos da abordagem "dirigida por força" são métodos intuitivos para criar um desenho de linhas retas em um grafo não direcionado. Eles são mais populares porque suas versões básicas são fáceis de entender e codificar [18].

Um algoritmo da abordagem "dirigida por força" simula um sistema de força definida em um grafo de entrada e saída em uma configuração de energia localmente mínima. Existem dois ingredientes desta abordagem:

- **Modelo de força:** Por exemplo, pode-se atribuir uma "mola" de comprimento natural l_{uv}, para cada par (u,v) de vértices. Pode-se escolher l_{uv} como sendo o número de arestas no caminho mais curto entre u e v. A mola segue a lei de Hooke, que é induzir a força de magnitude d_{uv} - l_{uv} em u, em que d_{uv} é proporcional à distância Euclidiana entre u e v.
- **Técnica para encontrar a configuração da energia localmente mínima:** Tais técnicas são usualmente o produto de análises numéricas ao invés de algoritmos combinatórios. Métodos simples e iterativos são comumente utilizados.

Experiências com algoritmos da abordagem "dirigida por força" mostram que eles podem produzir figuras bonitas de alguns dos bem conhecidos grafos em teoria dos grafos (tais como o esqueleto dos sólidos platônicos). Eles frequentemente resultam em desenhos simétricos e tendem a distribuir vértices uniformemente [18].

2.3.5. Metodologia dividir para conquistar

A abordagem dividir para conquistar é largamente adotada em desenho de grafos. A ideia básica é a seguinte: primeiramente, o grafo é dividido em subgrafos; em seguida os subgrafos são recursivamente desenhados; e por fim, o desenho de todo o grafo é obtido pela colagem dos desenhos do subgrafo. Para facilitar o entendimento de como *dividir para conquistar* pode ser usado em desenho de grafos, esboça-se um algoritmo para desenho de árvores binárias, proposto em [18]. O algoritmo consiste de dois passos principais:

- **O passo de atribuição em camadas é análogo ao da abordagem hierárquica**: Os vértices de uma árvore são atribuídos em camadas de maneira a minimizar a distância da raiz.
- **O passo dividir para conquistar:** Se a árvore consiste de um único vértice, então, trivialmente constrói-se o desenho na camada atribuída. Se a árvore está

vazia, então nada é feito. Senão, divide-se recursivamente o desenho da esquerda para a direita em subárvores. Na colagem, colocam-se os dois subdesenhos obtidos um próximo do outro, até que a distância horizontal entre eles seja 2. A raiz é posicionada em uma posição igual à metade entre as raízes das subárvores. Se uma das subárvores está vazia, a raiz é colocada a distância unitária da raiz da outra subárvore.

O algoritmo trabalha com a convenção de desenho planar de linhas retas no *grid* e leva em conta algumas estéticas. Por exemplo, se a largura do desenho é mantida a menor possível, subárvores isomórficas têm o mesmo desenho e subárvores simétricas têm o desenho da sua imagem espelhado [18].

Na próxima seção será apresentada a mais popular metodologia para desenho de grafos ortogonais no *grid*, a *topologia-forma-métrica*. Esta abordagem será melhor detalhada porque será utilizada como base para o desenvolvimento de algumas das novas metodologias desenvolvidas neste trabalho.

2.4. Metodologia topologia-forma-métrica

Desenhos ortogonais são intensivamente usados em aplicações da vida real. A abordagem *topologia-forma-métrica* [16], [17], [18], [19], foi desenvolvida para construir desenhos ortogonais no "*grid*" e permite tratamento homogêneo de amplo grupo de estéticas e restrições de desenho.

A idéia básica da abordagem é que um desenho ortogonal é caracterizado por três propriedades fundamentais, definidas em termos de classes de equivalência que foram estabelecidas entre desenhos ortogonais do mesmo grafo:

- **Topologia:** Dois desenhos ortogonais têm a mesma topologia se um puder ser obtido de outro por meio de deformações contínuas, que não alterem a sequência das arestas no contorno das faces do desenho.
- **Forma:** Dois desenhos ortogonais têm a mesma forma se eles têm a mesma topologia e um puder ser obtido do outro pela modificação dos comprimentos dos segmentos que compõem sua cadeia ortogonal, os quais representam suas arestas, sem modificar os ângulos formados entre eles.
- **Métrica:** Dois desenhos ortogonais têm a mesma métrica se eles são congruentes após uma translação e/ou rotação.

Cada uma das propriedades acima provê a descrição de um desenho que é um refinamento do anterior. Isto é, dois desenhos com a mesma métrica também têm a mesma forma e dois desenhos com a mesma forma também têm a mesma topologia. Note que os conceitos acima podem ser usados para caracterizar não somente desenhos ortogonais, mas, em geral, desenhos de polilinhas.

O relacionamento hierárquico entre a topologia, a forma e a métrica sugere a geração do desenho por passos, onde em cada passo uma representação intermediária é produzida [18].

Esta estratégia é ilustrada na *Figura 2.13* a seguir e será amplamente discutida neste capítulo.

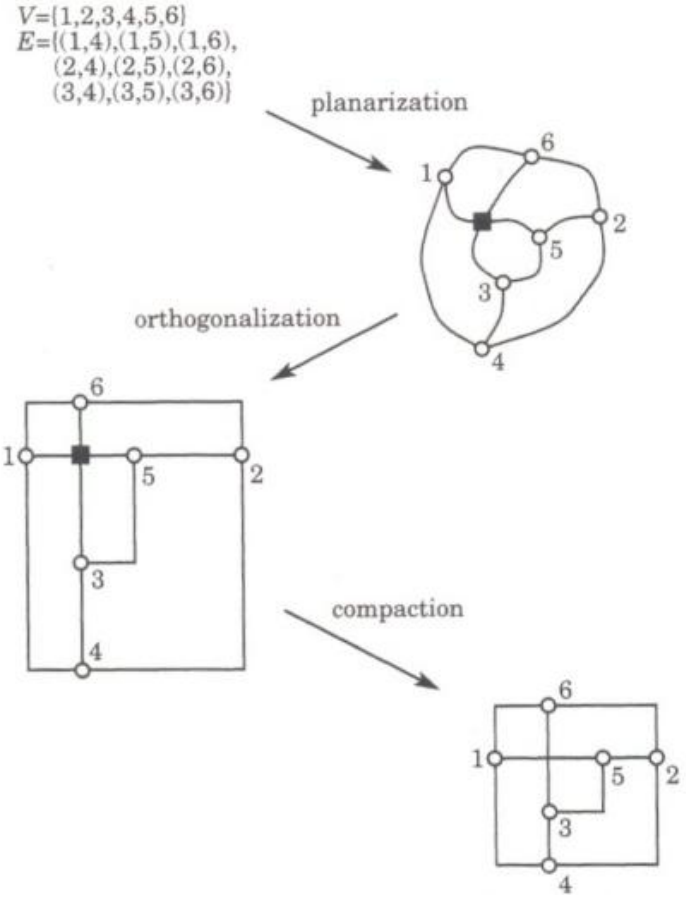


Figura 2.13 - Abordagem topologia-forma-métrica para desenho ortogonal no grid. O vértice fictício introduzido no passo de planarizaçao está representado por um quadrado [18].

- **Planarização:** Passo em que se determina a topologia do desenho, a qual é descrita por um embutimento planar [18], [60], [61]. Nessa fase, o problema é reduzir o número de cruzamentos entre as arestas, tanto quanto seja possível [18], [60], [62]. Esse problema tem sido intensivamente investigado na literatura. Um exemplo de implementação desta técnica na prática é o seguinte: um subgrafo planar maximal de um dado grafo é extraído [63] e as arestas não

planares são sucessivamente reinseridas, uma a uma, minimizando o número de cruzamentos causados em cada inserção. Cada cruzamento é representado por um vértice "fictício", de maneira que a topologia final seja planar [18].

- **Ortogonalização:** Dada uma topologia, o passo de *ortogonalização* determina a forma de um desenho [18]. Sua saída é uma representação ortogonal de um grafo. Em uma representação ortogonal, vértices não têm coordenadas e cada aresta (*u*,*v*) é equipada com uma lista de ângulos [18], [64]. Assim, as listas descrevem as dobras que a representação ortogonal da linha (*u*,*v*) terá no desenho final [22], [23], [65]. Será mostrado, nas próximas seções, como uma representação ortogonal pode ser obtida e como pode ser tratado o critério estético para *minimização do número de dobras* [66].
- **Compactação:** Dada uma representação ortogonal, o passo de *compactação* determina as coordenadas finais dos vértices e das dobras ao longo das arestas [18], [24], [67]. Nesta fase, o problema é usualmente o de produzir um desenho com o mínimo de área possível [24], [67]. Também os vértices "fictícios" introduzidos no passo de *planarização* são substituídos por representações gráficas dos cruzamentos [18].

Observa-se que a estratégia anterior determina uma ordem implícita de importância entre as estéticas. Isto é, o número de dobras da *forma* é afetado pela *topologia* e diferentes escolhas de topologias podem conduzir a formas com diferentes números de dobras. Consequentemente, a abordagem *topologia-forma-métrica* dá prioridade mais alta para a *minimização de cruzamentos* do que para a *minimização de dobras*, desde que se execute o passo da *planarização* antes do passo da *ortogonalização*. A topologia e a forma também afetam a área do desenho, portanto esta deverá ser tratada posteriormente. No passo de *compactação*, outras estéticas além da área podem ser levadas em conta, tais como a *minimização da soma dos comprimentos das arestas*, a *minimização do comprimento da maior aresta* ou a *minimização da razão de aspecto.*

Alguns tipos de restrições de desenho podem ser levados em conta na abordagem *topologia-forma-métrica*. As restrições podem ser subdivididas em restrições topológicas, formas e métricas. No passo de *planarização* pode-se, por exemplo, restringir que certos vértices fiquem na face externa do desenho ou previnam cruzamentos de arestas em um certo caminho. No passo de *ortogonalização* pode-se requerer que um dado caminho não contenha dobras ou pode-se impor sequências específicas de dobras em arestas específicas. No passo de *compactação*, pode-se restringir que certos vértices tenham coordenadas maiores ou menores que outros vértices. Naturalmente, devido à ordem dos passos, a abordagem dá mais alta prioridade para satisfazer às restrições topológicas em relação às restrições de forma e dá mais alta prioridade para satisfazer às restrições de forma em

relação às restrições de métrica. Nas próximas seções será abordada de forma aprofundada cada uma das três etapas da *topologia-forma-métrica* e seus algoritmos serão apresentados.

2.4.1. Planarização em grafos

Os problemas de testes de planaridade e construção de embutimentos planares de grafos têm sido extensivamente estudados e encontram aplicações diretas em uma variedade de áreas, incluindo *layout* de circuitos, gráficos, desenho auxiliado por computadores e desenho automático de grafos. Para a maioria das situações práticas é necessário conhecer previamente se um determinado grafo pode ser desenhado no plano, de tal forma que suas arestas só se intersectem nos pontos extremos. Um desenho desse tipo é chamado de desenho planar do grafo e um grafo para o qual existe um desenho planar é dito grafo planar [3], [4], [5], [8], [9], [10] (veja *Figura 2.14* para ilustração). Por exemplo, em projetos de circuitos integrados (VLSI), existe o interesse em se obter desenhos planares de grafos que representam circuitos elétricos.

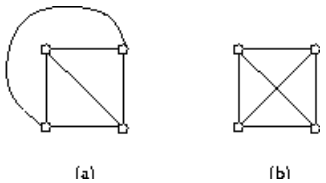


Figura 2.14 - (a) Um desenho planar de um grafo planar, (b) Um desenho do mesmo grafo que não é planar.

Uma caracterização muito simples de grafos planares foi dada por Kuratowski [68]. Ele demonstrou que todo grafo que não é planar contém uma subdivisão do grafo $K_{3,3}$ ou do grafo K_5 (veja *Figura 2.15*). Reciprocamente, nenhum grafo planar contém subdivisões de tais grafos. Uma subdivisão de $K_{3,3}$ ou K_5 contida num grafo é chamada de subgrafo de Kuratowski.

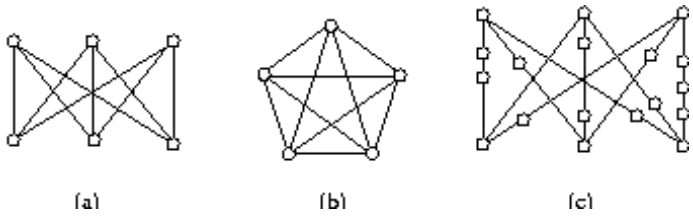


Figura 2.15 - (a) Grafo $K_{3,3}$, (b) Grafo K_5, (c) Uma subdivisão de $K_{3,3}$.

Apesar de elegante, a caracterização de Kuratowski não fornece um algoritmo muito prático para testes de planaridade. Além disto, não está claro como obter um desenho plano de um grafo planar através dessa caracterização.

O processo para obtenção de um grafo planar é chamado de *planarização*. A *planarização* é um dos passos da abordagem para desenho de grafos *topologia-forma-métrica* [18].

Uma possível estratégia para decidir se um grafo *G* é planar é construir incrementalmente um desenho plano de *G*. Se for possível completar a construção de um desenho de *G* no plano, então *G* é planar, caso contrário, idealmente, um subgrafo de Kuratowski é encontrado em *G*, justificando a não planaridade de *G*. O primeiro algoritmo incremental para o teste de planaridade foi proposto por Auslander e Parter [69]. Dado um grafo *G*, primeiramente, um circuito ou ciclo em *G* é encontrado e o grafo resultante da remoção desse ciclo é composto por componentes conexas. O algoritmo é chamado recursivamente para obter um desenho plano de cada uma dessas componentes. Depois, esses desenhos planos são combinados, se possível, para obter um desenho plano de *G*. Se não for possível combinar esses desenhos planos, um subgrafo de Kuratowski é encontrado em *G*. O algoritmo de Auslander e Parter continha um erro, pois entrava em *loop* infinito. Este erro foi corrigido por Goldstein que formulou o algoritmo corretamente utilizando iteração ao invés de recursão [60], [70], resultando no algoritmo de Auslander, Parter e Goldstein (APG). A complexidade de tempo desse algoritmo é cúbica no número de vértices $O(V^3)$.

Lempel, Even e Cederbaum [71] (LEC) apresentaram uma maneira alternativa de se construir um desenho plano incrementalmente. Nesta alternativa, começa-se com o desenho a partir de um único vértice e adiciona-se ao desenho todas as arestas incidentes a este vértice. Depois, adiciona-se um novo vértice incidente a uma das arestas já inseridas e todas as arestas incidentes a este novo vértice. O processo continua dessa maneira. Cada iteração do algoritmo começa com um desenho plano de parte do grafo e consiste em adicionar um novo vértice, bem como todas as arestas incidentes a este vértice, ao desenho planar corrente. O processo continua até que todo o grafo tenha sido desenhado no plano ou se tenha detectado que o grafo é não-planar. Este tipo de algoritmo incremental é chamado de *algoritmo de adição de vértices*. LEC não apresentou a implementação e nem a ordem de complexidade deste algoritmo. Tarjan [72] apresentou uma implementação do algoritmo de LEC que consome espaço O(V) e tempo $O(V^2)$, em que V é o número de vértices.

Hopcroft e Tarjan em [73] apresentaram um algoritmo que é uma variação do algoritmo de APG e que gasta tempo O(V*log*V) para os testes de planaridade. Na sequência, em 1974, Hopcroft e Tarjan [60] apresentaram o primeiro algoritmo linear (O(V)), uma variação mais simplificada do algoritmo de APG, para decidir se um dado grafo é planar.

Hopcroft e Tarjan não deram muitos detalhes de como o seu algoritmo de teste de planaridade pode ser estendido para construir de fato um desenho plano do grafo dado. Uma descrição completa da fase de construção de um desenho plano por esse algoritmo foi feita por Mehlhorn e Mutzel [74] e a respectiva implementação por Mutzel [75]. Ainda não se conhece uma implementação do algoritmo de Hopcroft e Tarjan que, em tempo linear,

devolve um certificado de não planaridade, ou seja, um subgrafo de Kuratowski de um grafo não-planar.

Em 1976, Booth e Lueker [76] apresentaram uma implementação do algoritmo de LEC que consome tempo linear (O(V)). Para isto, Booth e Lueker empregaram uma estrutura de dados chamada de PQ-árvore. Shih e Hsu [77] e Boyer e Myrvold [78] descreveram implementações similares para o algoritmo de LEC. Ambas gastam tempo linear em número de vértices e evitam o uso de PQ-árvores. Os dois algoritmos lineares em número de vértices para teste de planaridade mais discutidos na literatura são os algoritmos de APG e o de LEC. Canfield e Williamson [79] fazem uma comparação entre esses dois algoritmos argumentando que os dois, geralmente vistos como métodos bem diferentes, têm na realidade muitas semelhanças.

Tamassia [80] e Tamassia e Di Battista [18], [81], [82] desenvolveram algoritmos que são muito utilizados nos dias de hoje como excelentes técnicas para testes de planaridade e construção do embutimento de um grafo planar. Estes algoritmos são também baseados na estratégia de construção incremental. Neste caso, um desenho ou uma representação intermediária é construída pela adição de vértices e arestas de um grafo *G*. A cada inserção de aresta em um subgrafo ou *planarização G*'de *G*, um teste de planaridade é feito para ver se sua inserção mantém a planaridade de *G*'. Caso a inserção de uma aresta tenha provocado um cruzamento com outra aresta de *G*', ela é classificada como aresta não-planar. Finalmente as arestas não-planares são adicionadas uma a uma de forma a minimizar os cruzamentos. Isto é feito através da construção do dual do subgrafo *G*', onde o caminho mais curto (com menor número de arestas) no grafo dual de *G*' a partir das faces incidentes aos vértices extremos da aresta *e* é encontrado. A aresta *e* no grafo primal é traçada através deste caminho e um vértice fictício é adicionado no ponto onde ocorrer o cruzamento. Este procedimento é denominado *operação de planarização*. Esta estratégia será detalhada na próxima seção e o algoritmo para *planarização* incremental através da construção do grafo dual será apresentado.

A *Tabela 2.1* resume as complexidades de tempo em relação ao número de vértices para as implementações dos principais algoritmos para *planarização* citados.

Tabela 2.1 –Algoritmos de planaridade e algumas de suas implementações e complexidades.

Algoritmo	Implementação	Complexidade de tempo
Auslander, Parter e Gosdstein (APG)	Original [69,70] Hopcroft e Tarjan [73] Hopcroft e Tarjan [60]	$O(V^3)$ O(VlogV) O(V)
Lempel, Even e Cederbaun (LEC)	Original [68] e [72] Booth e Luker [76] Boyer e Mirvold [78] Shih e Hsu [77]	$O(V^2)$ O(V) O(V) O(V)
Incremental (dual)	Roberto Tamassia *et al*[18]	O(V)

2.4.2. Planarização incremental

Uma *operação de planarização* [18] consiste em adicionar um novo vértice *x* no lugar do cruzamento e substituir os pares (*a*,*b*), (*c*,*d*) de arestas não-adjacentes pelas quatro arestas (*a*,*x*), (*b*,*x*), (*c*,*x*) e (*d*,*x*) que se formam. Veja *Figura 2.16* para ilustração. Intuitivamente, a operação de *planarização* adiciona um vértice fictício *x*, onde as arestas (*a*,*b*) e (*c*,*d*) se cruzam.

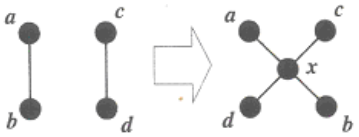


Figura 2.16 - Operação de planarização [18].

Uma planarizaçao *G'* de um grafo *G* é um grafo planar obtido a partir do grafo *G* pela sequência de operações de *planarização*.

Será apresentado o algoritmo *planarize* [18], onde algumas restrições de natureza topológica podem ser suportadas, inclusive:

- Prevenir cruzamentos de arestas.
- Colocar os vértices no limite externo.

O algoritmo *planarize* é um algoritmo simples para encontrar a *planarização* de um grafo e gasta tempo *O*(V) em número de vértices. Ele utiliza um algoritmo para encontrar um subgrafo maximal planar como uma subrotina. O indicado na literatura como melhor algoritmo para obtenção do subgrafo maximal planar é descrito em [83]. Veja a *Figura 2.17* para ilustração e melhor compreensão do algoritmo *planarize*.

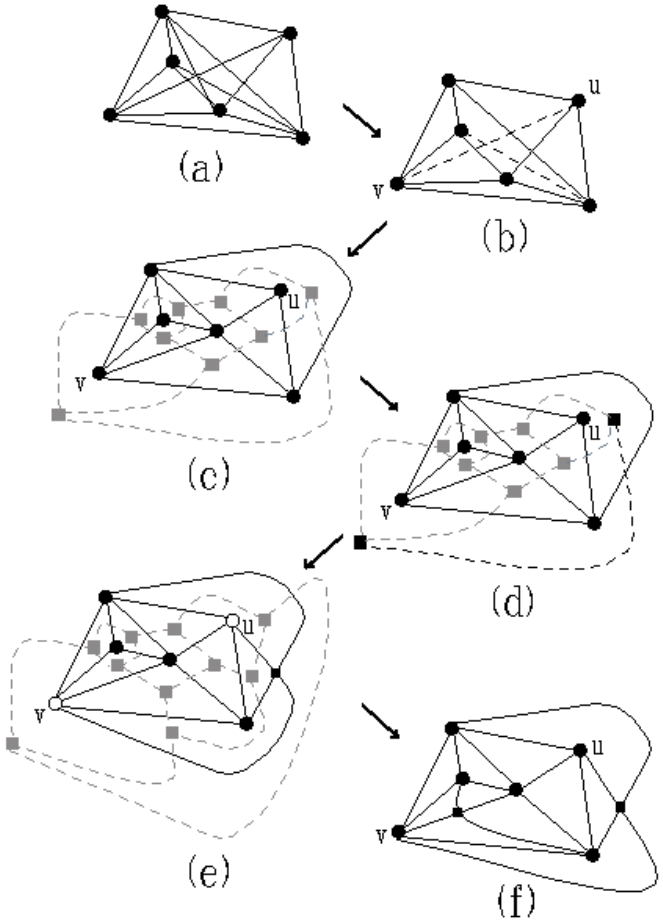


Figura 2.17 - Método de planarização simples: (a) grafo inicial; (b) particionamento das arestas em planares (contínuas) e não-planares (pontilhadas); (c) grafo dual (sombreado), usado para detectar as arestas não-planares; (d) caminho mais curto para aresta não-planar no grafo dual; (e) embutimento planar e grafo dual após a inserção de uma aresta não-planar; (e) grafo final planarizado [18].

Algoritmo Planarize

Entrada: Grafo *G*.

Saída: Planarização *G'* de *G*.

1. Calcule o subgrafo maximal planar *S* de um grafo de entrada *G* e particione as arestas em "planares" e "não-planares", como segue (veja a *Figura 2.17 b*):
   - Inicialize com subgrafo *G*' constituído somente pelos vértices de *G*, sem nenhuma aresta.
   - Para cada aresta *e* de *G*, se o grafo obtido pela adição de *e* em *G*' for planar, então adicione *e* em *G*' e a classifique como "planar", senão a rejeite e a classifique como "não-planar".
2. Construa um embutimento planar de um subgrafo planar *G*' e o grafo dual de *S* (veja a *Figura 2.17 c*).

3. Adicione a *G*' as arestas "não-planares", uma de cada vez, e a cada vez minimizando o número de cruzamentos. Isto é feito conforme abaixo para a aresta "não-planar" (*u*,*v*):

- Encontre um caminho mais curto (menor número de arestas) no grafo dual do embutimento *G*' a partir das faces incidentes a *u* para as faces incidentes a *v* (veja a *Figura 2.17 d*).
- Adicione a aresta "não-planar" e atualize *G*' e seu dual (veja *Figura 2.17 e*).
- Aos cruzamentos inevitáveis adicione um vértice falso em seu lugar. O número de vértices falsos adicionados representa o número de cruzamentos do grafo.

2.4.3. Ortogonalização de grafos planares

Dado um embutimento planar Γ de um grafo *G*, a *ortogonalização* é responsável por criar uma *representação ortogonal* de Γ, denominada ***H***. A *ortogonalização* é uma das etapas da abordagem *topologia-forma-métrica* para desenho de grafos [6], [7], [18], [19], [51]. Nesta etapa, o passo de *ortogonalização* recebe como entrada um embutimento planar Γ e devolve a *representação ortogonal* ***H*** de Γ. Ao final desta etapa, os vértices não possuem coordenadas, mas cada aresta contém uma lista de ângulos que descrevem para que lado serão suas dobras. A abordagem *topologia-forma-métrica* trata criteriosamente as propriedades estéticas do grafo, para garantir, entre outras, a questão relacionada com a legibilidade no desenho dos grafos. Um dos critérios estéticos, especialmente importante e já citado anteriormente, está relacionado com o número total de dobras em um desenho ortogonal. É naturalmente desejável a *minimização do número total de dobras* ao longo das arestas do desenho, de maneira a prover maior legibilidade ao desenho. Para que seja possível resolver o problema de *minimização do número total de dobras* ao longo das arestas, o problema da *ortogonalização* é transformado em um problema de fluxo de custo mínimo em rede e é resolvido por esta abordagem [17], [18] e [84]. Outra forma de se resolver o problema é a transformação deste em um problema de programação linear inteira [30], [31], [32], [33] e resolvê-lo por técnicas desta natureza.

Nas próximas seções serão apresentadas definições importantes para a compreensão da etapa de *ortogonalização* de um grafo planar como: ângulos em desenhos ortogonais; representações ortogonais; fluxo em rede, entre outras.

2.4.4. Ângulos em desenho ortogonal

Seja ψ um desenho ortogonal planar de um embutimento planar Γ de um grafo *G*.
Existem dois tipos de ângulos em ψ:

- Ângulos formados por duas arestas incidentes a um vértice comum são chamados de *ângulos-vértice*.
- Ângulos formados por dobras (que são ângulos formados por segmentos consecutivos da mesma aresta) são chamados de *ângulos-dobra*.
- Ângulos formados nos vértices ou dobras dentro da face são denominados ângulos-faces.

As *Figuras 2.18 e 2.19* ilustram *ângulos-vértices* e *ângulos-face*, respectivamente.

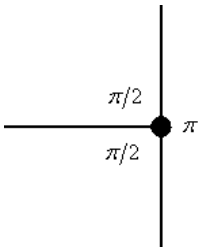


Figura 2.18 - Ângulos em volta do vértice (ângulo-vértice) em um desenho ortogonal planar.

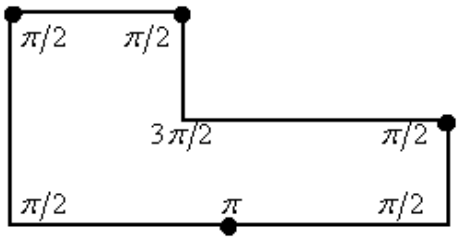


Figura 2.19 - Ângulos dentro da face (chamados ângulos-face) em um desenho ortogonal planar.

As seguintes propriedades são imediatas:

Lema 2.1 – *Em um desenho ortogonal planar, a soma das medidas dos ângulos-vértices em torno de um vértice é igual a* 2π .

Lema 2.2 – *Seja f uma face interna de um desenho ortogonal planar. A soma das medidas dos ângulos-vértice e ângulos-dobra dentro da face f é igual a* $\pi(p-2)$, *onde p é o número total de tais ângulos. Se f é a face externa, então a soma acima é igual a* $\pi(p+2)$.

Utilizando o exemplo da *Figura 2.19,* será verificado o **Lema 2.2**:

Soma das medidas dos ângulos vértices (*av*) = $\frac{\pi}{2}+\frac{\pi}{2}+\frac{\pi}{2}+\pi=\frac{5\pi}{2}$;

Soma das medidas dos ângulos dobras (*ad*) = $\frac{\pi}{2}+\frac{\pi}{2}+\frac{3\pi}{2}=\frac{5\pi}{2}$;

$$av + ad = \frac{5\pi}{2} + \frac{5\pi}{2} = \frac{10\pi}{2} = 5\pi\,; \tag{2.1}$$

O número total dos ângulos da face interna do desenho ortogonal (*p)* = 7

Logo $\pi(p-2) \; -> \pi(7-2) = 5\pi$ (2.2)

Como (2.1) = (2.2), o **Lema 2.2** é verificado. De forma análoga pode ser feito para a face externa.

2.4.5. Representação ortogonal

Introduz-se o conceito de representação ortogonal, o qual captura a noção de "forma ortogonal" de um desenho ortogonal planar, considerando os ângulos e não o tamanho de suas arestas.

Seja Γ um embutimento planar de um grafo *G*, cujos vértices sejam de grau no máximo quatro. Denota-se por a(*f*) o número total de *ângulos-vértices* dentro da face *f* de Γ. Se *G* for biconectado, então a(*f*) é igual ao número de vértices (arestas) de *f*. Para cada aresta *e* (não-direcionada) de *G* cujos pontos finais sejam *u* e *v*, as duas possíveis orientações (*u*,*v*) e (*v*,*u*) da aresta *e* são chamadas semi-arestas. Diz-se que uma semi-aresta está no sentido anti-horário com relação a sua face *f* se *f* está do lado esquerdo quando a semi-aresta na direção de sua orientação é atravessada. Denota-se por *D*(*v*) o conjunto de semi-arestas que se iniciam no vértice *v*, e por *D*(*f*) o conjunto de semi-arestas da face *f* que a percorrem no sentido anti-horário.

Dado um desenho ortogonal planar ψ de um embutimento planar Γ do grafo *G*, os valores α e β associados às semi-arestas de *G* são definidos como segue:

- $\alpha(u,v) \cdot \pi/2$ é o ângulo no vértice *u* formado pelo primeiro segmento da semi-aresta (*u*,*v*) e o próximo segmento no sentido anti-horário em torno de *u*. No exemplo da *Figura 2.20*, $\alpha(u,v)$ é dado pelo ângulo formado pela aresta que liga *u* a *v* e a primeira aresta em torno de *u* no sentido anti-horário, que, no caso, seria a aresta (*u*,*w*). Assim $\alpha(u,v)$ = 1.
- $\beta(u,v)$ é o número de dobras ao longo da semi-aresta (*u*,*v*) com o ângulo $\pi/2$, no lado esquerdo da mesma. No exemplo da *Figura 2.20*, seguindo a aresta (*u*,*v*), no sentido anti-horário, são encontradas duas dobras que possuem do seu lado esquerdo um ângulo $\pi/2$ em cada, portanto $\beta(u,v)$ = 2.

Veja a *Figura 2.20* para ilustração.

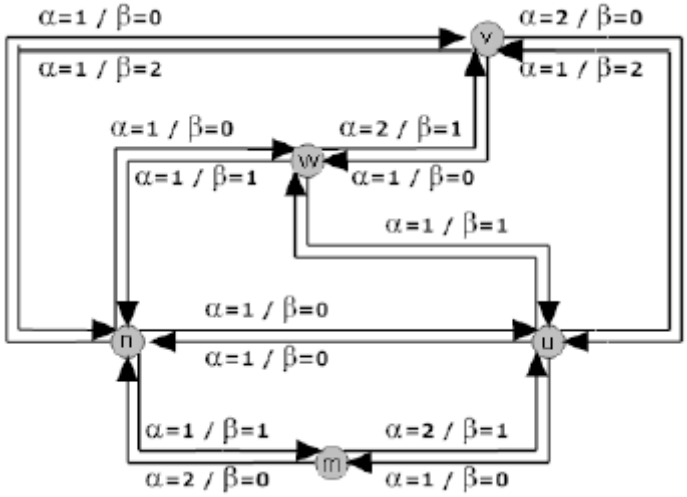


Figura 2.20 - Valores de α *e* β *[6].*

Uma representação ortogonal de *G* descreve uma classe de equivalência do desenho ortogonal planar de *G* com "forma similar", que tem os mesmos valores de α e β, associados às semi-arestas de *G*. De maneira mais formal, pode-se dizer que uma *representação ortogonal* ***H*** de *G* é uma atribuição de valores inteiros $\alpha(u,v)$ e $\beta(u,v)$, para cada semi-aresta (u,v) de *G*, que satisfaça as seguintes propriedades (**Lema 2.1** e **Lema 2.2**):

- $1 \leq \alpha(u,v) \leq 4$
- $\beta(u,v) \geq 0$
- Para cada vértice *u*, a soma de $\alpha(u,v)$ sobre todas as semi-arestas orientadas fora de *u* é igual a quatro, que é dado por:

$$\sum_{(u,v)\in D(u)} \alpha(u,v) = 4 \tag{2.3}$$

- Para cada face interna *f*, a soma de $\alpha(u,v) + \beta(v,u) - \beta(u,v)$ sobre todas as semi-arestas(u,v) de *f*, no sentido anti-horário, é igual a 2a(*f*) - 4, que é dado por:

$$\sum_{(u,v)\in D(f)} \alpha(u,v) + \beta(v,u) - \beta(u,v) = 2a(f) - 4 \tag{2.4}$$

- Para a face externa *h*, a soma acima é igual a 2a(*h*) + 4, que é dado por:

$$\sum_{(u,v)\in D(h)} \alpha(u,v) + \beta(v,u) - \beta(u,v) = 2a(h) + 4 \tag{2.5}$$

A *Figura 2.21* mostra três desenhos ortogonais planares de um grafo com a mesma *representação ortogonal* ***H***. Pode-se notar que ***H*** descreve a "forma" de um desenho ortogonal, dependendo da permutação da ordem das dobras ao longo das arestas. Em particular, dois desenhos ortogonais, com a mesma *representação ortogonal*, têm o mesmo número de dobras (o mesmo para a soma dos valores de β sobre todas as semi-arestas).

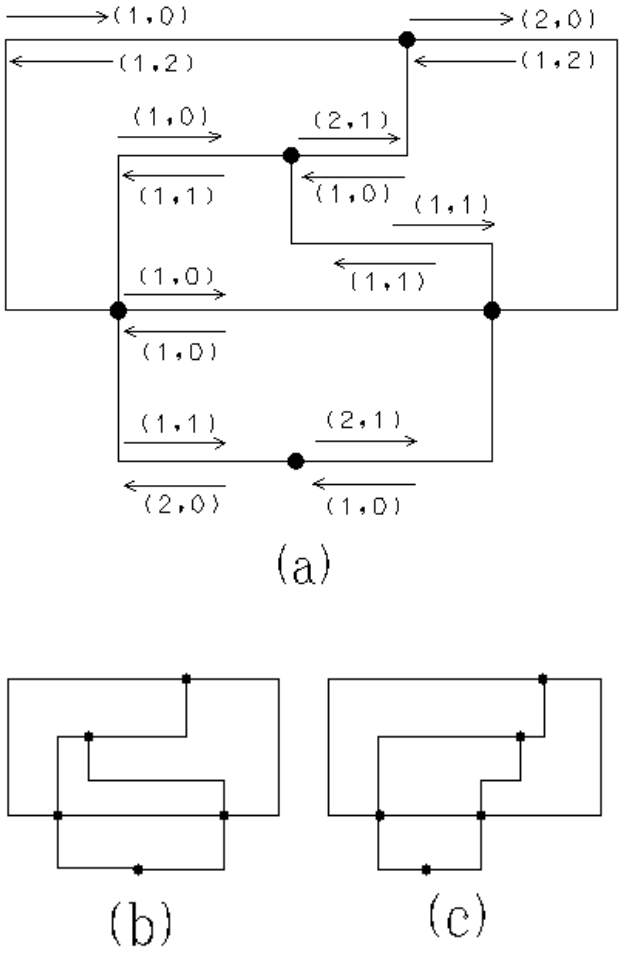


Figura 2.21 - Três desenhos ortogonais com a mesma representação ortogonal. Cada semi-aresta(u,v) de um desenho na figura (a) é nomeado com o par (α (u,v), β (u,v)).

2.4.6. Fluxo em rede e desenho ortogonal de grafos

Técnicas de fluxo em rede podem ser utilizadas para resolver uma variedade de problemas de desenho de grafos planares [6], [7], [17], [18], [19], [49], [64]. Os ângulos formados pelas arestas, nos vértices e nas dobras, satisfazem as propriedades geométricas que podem ser expressas por um modelo de fluxo. Será analisada a metodologia para desenho de grafos apresentada por Tamassia em [17] e discutida amplamente em [18], [19] e [54]. Esta tem como base as técnicas de fluxo de rede, a qual constrói um desenho planar ortogonal a partir de um embutimento planar de um grafo com o número mínimo de dobras. Este problema pode ser modelado como um problema de fluxo de custo mínimo em rede, ou

seja, um fluxo em rede derivado de um grafo e seu embutimento. Neste caso, cada unidade de fluxo corresponde a um ângulo $\pi / 2$, os vértices são *produtores* de quatro unidades de fluxo, as faces são *consumidoras* de uma quantidade de fluxo proporcional ao número de ângulos em seu interior e cada dobra transfere uma unidade de fluxo através de sua face incidente. Atribuindo unidades de custo ao fluxo associado às dobras, um desenho com um número mínimo de dobras corresponde ao fluxo de custo mínimo. Isto lida com algoritmos de tempo quadrático e espaço linear em número de vértices para a *minimização do número de dobras*. Esta técnica foi primeiro apresentada em [17], com variações, refinamentos e extensões dadas em [19], [49], [64]. Hoje, já é possível ortogonalizar um grafo planar em tempo $O(V^{7/4}\log V)$ e espaço O(V) [84]. A *minimização do número de dobras* em todos os possíveis embutimentos de um grafo planar [15] é um problema da classe NP-difícil. As técnicas discutidas nesta seção podem também ser utilizadas para desenhos de grafos não planares em geral, através de um passo de *planarização* [18] aplicado ao grafo não-planar.

Um modelo de fluxo em rede para o problema de construção de um desenho ortogonal planar de um embutimento planar Γ é mostrado. Este modelo mostra os ângulos como sendo artigos que são *produzidos* pelos vértices e transportados entre as faces por arestas, através de suas dobras, e, eventualmente, são *consumidos* pelas faces. Consequentemente, os vértices e as faces do grafo são os nós da rede. Como o desenho deve ser ortogonal, todos os ângulos devem possuir medida do tipo $k \cdot \pi/2$, na qual $1 \leq k \leq 4$. Uma unidade de fluxo corresponde a um ângulo de $\pi/2$.

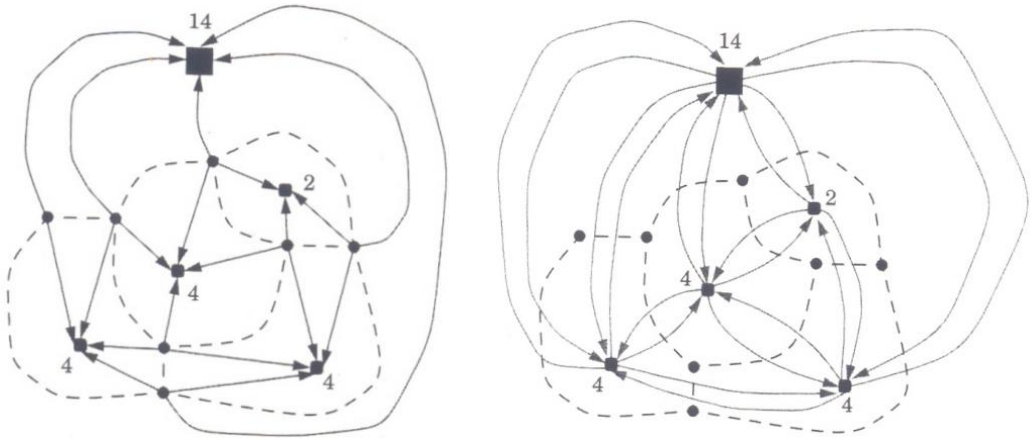
Associa-se uma rede $\mathcal{N}$ cujos nós têm *requisições* e *demandas* e que cada arco tem um limite inferior λ, uma capacidade μ, e um custo χ, com um embutimento planar de um grafo *G* como mostrado na *Figura* 2.*22*.

Denomina-se *nó-vértice* aquele nó que se encontra nos vértices da representação ortogonal e *nó-face* aquele nó que representa cada face da representação ***H***. Na rede $\mathcal{N}$, o *nó-vértice* é representado por uma bolinha preta e o *nó-face* por um quadrado preto.

- Os nós de $\mathcal{N}$ são os vértices das faces de *G*;
- Um *nó-vértice* de $\mathcal{N}$ produz o fluxo $\sigma(v) = 4$
- Um *nó-face* de $\mathcal{N}$ consome fluxo $\sigma(f) = 2a(f) - 4$ se *f* é um nó interno da face e fluxo $\sigma(h) = 2a(h) + 4$ se *h* é a face externa.
- Para cada semi-aresta (*u*,*v*) de *G*, com faces *f* e *g* na sua esquerda e direita respectivamente, $\mathcal{N}$ tem dois arcos (*u*,*f*) e (*f*,*g*), onde:
    - Arco(*u*,*f*) tem limite inferior $\lambda(u,v) = 1$, capacidade $\mu(u,v) = 4$ e custo $\chi(u,v) = 0$. (ver *Figura 2.22.a*)
    - Arco (*f*,*g*) tem limite inferior $\lambda(u,v) = 0$, capacidade $\mu(u,v) = +\infty$ e custo $\chi(u,v) = 1$. (ver *Figura 2.22b*)

Em palavras mais simples, as definições de fluxo da rede $\mathcal{N}$ são as seguintes:

- O fluxo no arco(*u*,*f*), associado a semi-aresta(*u*,*v*), representa a quantidade α (*u*,*v*), que é a medida do ângulo formado pelo vértice *u* dentro da face *f*. O limite inferior e a capacidade indicam que tais ângulos devem ser, no mínimo, $\pi/2$ e, no máximo, 2π. O custo é zero, pois o ângulo está no vértice e não na dobra.
- O fluxo no arco(*f*,*g*), associado à semi-aresta (*u*,*v*), representa a quantidade β (*u*,*v*), que, por sua vez, representa o número de dobras com ângulo igual a $\pi/2$ na face *f*, ao longo da aresta, entre as faces *f* e *g*. O limite inferior e a capacidade indicam que tais números devem ser não-negativos e podem ser ilimitados. O custo vale 1, desde que cada unidade de fluxo em tais arcos corresponda a uma dobra.
- A conservação de fluxo no *nó-vértice* representa a propriedade expressa pelo **Lema 2.1**.
- A conservação do fluxo no *nó-face* representa a propriedade expressa pelo **Lema 2.2**.



(a) (b)

Figura 2.22 - Rede $\mathcal{N}$ associada a um embutimento planar do grafo G (dividida em dois desenhos apenas para facilitar o entendimento). A Cada nó-face f de $\mathcal{N}$ é atribuído um fluxo σ (f) consumido: a) arcos de $\mathcal{N}$ dos nós-vértices para os nós-faces; b) arcos de $\mathcal{N}$entre os nós-faces [18].

A *Figura 2.23* mostra a correspondência entre o fluxo na rede $\mathcal{N}$ e a representação ortogonal do grafo *G*. Nesta figura, pode-se observar que cada *nó-vertice* produz uma unidade de fluxo a qual é consumida pelo *nó-face* na face interna. Cada dobra produz uma unidade de fluxo que será consumida pelo *nó-face* na face interna se a medida do *ângulo-dobra* for $3\pi/2$ e estiver dentro da face interna ou será consumida pela face externa se a medida do *ângulo-dobra* for $3\pi/2$ e estiver na face externa.

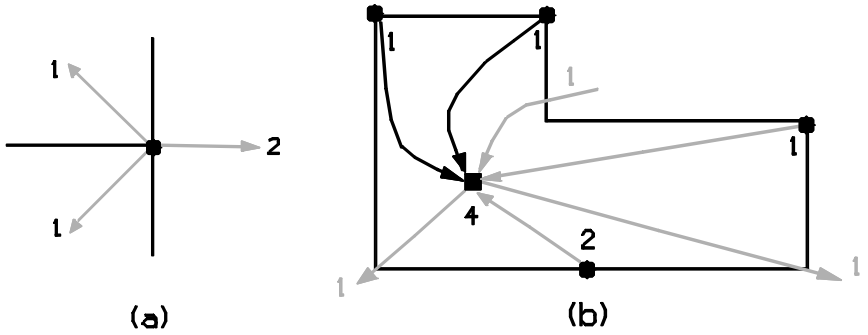


Figura 2.23 - Correspondência entre o fluxo na rede $\mathcal{N}$ e a representação ortogonal de G: a) nó-vértice; b) nó-face.

A quantidade total de fluxo produzida pelos *nós-vértices* é igual à quantidade total de fluxo consumida pelos *nós-faces*. Portanto, a conservação do fluxo na rede $\mathcal{N}$, associada ao embutimento planar de *G*, é descrita pela *Equação* (*2.6*) (as faces internas e a face externa são consideradas dentro de um único somatório).

$$\sum_{v}\sigma(v)-\sum_{f}\sigma(f)=4\bullet v-(\sum_{f}(2a(f)-4))-8 \qquad (2.6)$$

O custo de um fluxo ϕ em uma rede $\mathcal{N}$, construída conforme a descrição anterior, equivale ao número total de dobras de uma representação ortogonal ***H*** de *G*. Além disso, o fluxo de um arco entre *nós-vértices* e *nós-faces* $\phi(u,f)$ corresponde ao valor $\alpha(u,v)$ da aresta associada em ***H***, enquanto o fluxo de um arco entre *nós-faces* $\phi(f,g)$ corresponde ao valor $\beta(u,v)$ dessa mesma aresta. Com base nessas ideias, o algoritmo ***Ortogonaliza*** descreve a etapa de *ortogonalização*.

Algoritmo Ortogonaliza

Entrada: Embutimento planar de *G* com grau no máximo 4.
Saída Representação ortogonal ***H*** com o menor número de dobras.
1. Construa a rede $\mathcal{N}$ de fluxo ϕ associada a *G*.
2. Calcule o fluxo ϕ de custo mínimo em $\mathcal{N}$.
3. Construa uma *representação ortogonal* ***H***, através da equivalência de fluxos nos arcos da rede e valores de α e β.

2.4.7. Representação ortogonal e técnicas de Programação Linear Inteira (PLI)

As etapas *planarização* e *compactação* da *topologia-forma-métrica* [17], [19], [67], [85], [86], [87] são tradicionalmente implementadas através de heurísticas [17]. Na década de 90 foram desenvolvidos métodos mais avançados relacionados com estrutura de dados para grafos [65] e otimização [23]. Assim, foi possível resolver a etapa de *ortogonalização* de maneira eficiente, minimizando o número de dobras do desenho. Esta etapa é originalmente baseada na abordagem do fluxo de custo mínimo [17], [18], como apresentado na *Seção 2.4.6*. Nesta seção será apresentado um método para desenho ortogonal considerando uma larga variedade de restrições de desenho, em que se tenha uma maior flexibilidade no desenho de grafos de acordo com os requisitos dos usuários [22], [23].

Geralmente se distingue entre *restrições de arestas* e *restrições de vértices*. *Restrições de arestas* requerem propriedades especiais para arestas, tais como: execução em uma direção pré-estabelecida (*Upward* - "desenhos para cima" é um exemplo de restrições em relação às arestas), ou ter um tamanho predeterminado (restrição de tamanho das arestas) ou estar conectada a um determinado vértice do lado definido pelo usuário (restrição de lado), ou ainda, em algum lugar do vértice, definido pelo usuário. *Restrições de vértices* incluem condições em relação às posições dos vértices.

Técnicas de PLI podem permitir o tratamento de um vasto número de restrições bem como de critérios estéticos. Esta abordagem será tratada na próxima seção.

2.4.8. Ortogonalização utilizando técnicas de PLI

A análise do problema de utilização de técnicas de PLI estará restrita apenas a grafos planares. Esta abordagem pode ser estendida a grafos não-planares, através da aplicação da etapa de *planarização*, e desta maneira funcionará perfeitamente para a abordagem (PLI) [22], [23] que será mostrada a seguir.

Como entrada para o algoritmo proposto, assume-se um embutimento planar $\varepsilon = \cup_{v \in V} \varepsilon(v)$, de um grafo G (V,E), com vértices V e arestas E, dados pela ordenação cíclica no sentido horário $\varepsilon(v)$ de uma aresta incidente, em torno de cada vértice v. Cada aresta $e = (v,w) \in E$ aparece duas vezes em ε, ou seja, uma aresta (v,w) em $\varepsilon(v)$ é também representada pela aresta (w,v) em $\varepsilon(w)$. Além disto, a face externa do grafo planar é prescrita. A partir desse conjunto de dados, o conjunto de faces F pode ser determinado.

Seja F_{in} o conjunto de faces internas e F_{out} o conjunto de faces externas (o qual é sempre composto por uma única face). Cada face *f* de *F* é armazenada como uma lista de arestas ordenadas, que definem *f*. Cada semi-aresta *e* em uma face *f* é direcionada de maneira que a face descrita está do lado direito da aresta *e* (considerando o sentido horário).

Em uma *representação ortogonal*, cada aresta é especificada pela direção de seus primeiros segmentos e pela sequência das dobras seguintes. Quando o algoritmo calcula estes dados, ele deve, neste momento, levar em consideração as funções de restrições. Será reformulado, então, o modelo do problema baseado na abordagem de fluxo de custo mínimo [17], [18], [19] por um sistema com base em equações e inequações lineares. O modelo equivalente ao problema apresentado no início do capítulo (com base em fluxo de custo mínimo em rede) é simples. Observa-se que o grau máximo dos vértices é 4. Tem-se, com isto, dois tipos de variáveis no sistema: O primeiro tipo está relacionado com o número de dobras. Nele, para cada aresta $(u,v) \in \varepsilon$, existe uma variável $r_{(u,v)}$, que conta o número de dobras com ângulo $\pi/2$ (dobras para direita), e uma variável $l_{(u,v)}$, que conta o número de dobras com ângulo $3\pi/2$ (dobras para esquerda), ao longo daquela aresta na direção *u*-*v*. As variáveis $l_{(v,u)}$ e $r_{(v,u)}$ contam o número de dobras na direção inversa *v*-*u*. Note que as dobras de $3\pi/2$, em uma dada direção, são as dobras de $\pi/2$ na direção inversa. Fica fácil observar que tanto $l_{(u,v)}$ quanto $r_{(u,v)}$ valem zero em um desenho sem dobras [17], [23]. Para obter um desenho de dobras-mínimas, minimiza-se a soma das variáveis $l_{(u,v)}$ e $r_{(u,v)}$. O segundo tipo de variável corresponde aos ângulos de vértices. Para cada $e = (u,v) \in \varepsilon$ existe uma variável $a_{(u,v)}$ que denota o ângulo entre *e* e seu predecessor cíclico em $\varepsilon(v)$ no vértice *v*. O valor *r* de uma variável $a_{(u,v)}$ corresponde ao valor de $r \cdot \pi/2$ do ângulo correspondente. Todos os ângulos crescentes têm que ser múltiplos de $\pi/2$. Assim, vê-se explicitamente que as variáveis têm valores (*r*) inteiros (1, 2, 3 e 4). Desta forma, chega-se ao seguinte problema de Programação Linear Inteira (PLI):

$$\min \sum_{(v,w)\in E} (l_{(v,w)} + r_{(v,w)}) \tag{2.7}$$

Sujeito a:

$$\sum_{(v,w)\in \varepsilon(v)} a_{(v,w)} = 4 \qquad \forall v \in V \tag{2.8}$$

$$\sum_{(v,w)\in f} (a_{(v,w)} + l_{(v,w)} - r_{(v,w)}) = \begin{cases} 2k-4 & f \in F_{in},\ |f| = k \\ 2k+4 & f \in F_{out},\ |f| = k \end{cases} \qquad \forall f \in F \tag{2.9}$$

$$l_{(v,w)} = r_{(w,v)} \qquad \forall (v,w) \in \varepsilon \tag{2.10}$$

$$l_{(v,w)},\ r_{(v,w)} \in N \qquad \forall (v,w) \in \varepsilon \tag{2.11}$$

$$a_{(v,w)} \in \{1,\ldots,4\} \qquad \forall (v,w) \in \varepsilon \tag{2.12}$$

Em [19] é mostrado que estas condições são suficientes para gerar desenhos ortogonais corretos.

As abordagens baseadas em PLI são consideradas as mais eficientes para tratar o problema de desenhos ortogonais em grafos planares [22], [23], [24], [25] e [26], assim, elas serão utilizadas no desenvolvimento de parte deste trabalho.

2.4.9. Compactação de desenhos ortogonais de grafos planares

A *compactação* é a terceira etapa da abordagem *topologia-forma-métrica* [18], na qual se efetua a construção de um desenho ortogonal no *grid* a partir de uma dada representação ortogonal ***H*** de um grafo *G* (saída da etapa de *ortogonalização*).

Nesta etapa é necessário efetuar a atribuição de *tamanho* aos segmentos das arestas da representação ortogonal ***H***, de maneira que não haja cruzamentos ou sobreposições entre vértices e arestas. É necessário garantir que a área do desenho seja a menor possível, considerando que são utilizados somente valores inteiros para o tamanho dos segmentos. Deste fato é que vem o nome "*compactação*".

A abordagem proposta por [17], [19] e discutida em [6], [14], [18], [64], para tratar o problema de *compactação*, considerando a abordagem *topologia-forma-métrica*, consiste em, a partir de uma *representação ortogonal* ***H,*** construir-se o desenho ortogonal no *grid* ψ, atribuindo tamanhos inteiros aos segmentos de arestas. Neste momento, é necessário tratar os critérios estéticos relacionados com esta etapa e que são relevantes para a convenção de desenhos ortogonais no *grid*: *minimização da soma total do comprimento das arestas, minimização da área do desenho, minimização da razão de aspecto e a minimização do tamanho da maior aresta*, sem com isto afetar a topologia e a forma do desenho.

Em desenhos ortogonais no *grid* ψ, cada aresta é representada como uma cadeia de segmentos horizontais e verticais, e, além disso, os vértices e dobras são colocados nos pontos do *grid*. Dois desenhos ortogonais têm a mesma forma, se um puder ser obtido do outro pela modificação dos tamanhos dos segmentos horizontais e verticais, sem modificar os ângulos formados entre eles. Assim, o problema da *compactação* trata separadamente dos segmentos horizontais e dos segmentos verticais. A maioria dos algoritmos utilizados, para a solução deste tipo de problema, teve suas raízes em desenho de circuitos integrados (VLSI) e foram adaptados para resolver o problema de *compactação* em desenho de grafos [22].

Em desenho de grafos, heurísticas têm sido utilizadas para *compactação* de desenhos ortogonais no *grid*. Tamassia *et. al* [18] sugeriram a utilização de refinamentos na forma do desenho obtendo uma forma com faces retangulares pela introdução de arestas artificiais. Se todas as faces são retangulares, o problema da *compactação* pode ser resolvido em tempo polinomial usando algoritmos relacionados com fluxo de custo mínimo em redes. Entretanto, de maneira geral, esta solução está longe de ser uma solução ótima para o grafo original, ou seja, sem as arestas artificiais. Outras técnicas, bem como comparações entre elas, são também mostradas em [24], [25]. A técnica com base em programação linear inteira (PLI) desenvolvida em [24], [25], para solução do problema de compactação de maneira ótima e que mostrou resultados superiores as demais técnicas analisadas, será discutida neste capítulo e servirá de base para o desenvolvimento de uma das metodologias propostas neste trabalho.

Nas próximas seções serão discutidas as metodologias com base no fluxo de custo mínimo em rede. Esta metodologia exige que o desenho possua apenas faces retangulares. O seu algoritmo será mostrado. Na sequência, serão mostradas as metodologias de compactação uni-dimensional e bi-dimensional de forma a prover subsídios à compreensão da técnica que utiliza a abordagem dos segmentos de maneira a permitir o tratamento do problema de compactação como um problema de programação linear inteira (PLI). Esta técnica será mostrada em seções posteriores e servirá de base para o desenvolvimento da abordagem proposta.

2.4.10. Representação ortogonal com faces retangulares

A *Figura 2.24* ilustra uma representação ortogonal ***H*** para um grafo planar *G*, em que todas as faces da representação ortogonal são retangulares.

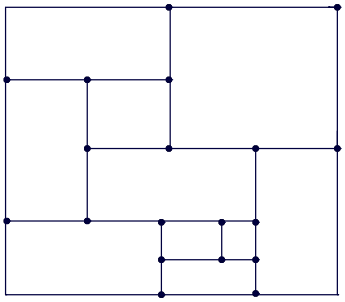


*Figura 2.24 - Representação ortogonal **H** com faces retangulares.*

A representação ortogonal ***H*** tem no máximo 4 dobras, as quais se encontram nos "cantos" da face externa. Qualquer outra dobra fica incompatível com o requisito que as faces sejam

retangulares. Consequentemente, aos segmentos de ***H*** correspondem as arestas de *G*, exceto, possivelmente, para no máximo oito segmentos incidentes às dobras da face externa (4 dobras). Formalmente o problema pode ser descrito da seguinte maneira:
Uma face f tem forma retangular se:

- $\alpha(u,v) \leq 2$
- $\beta(u,v) = 0$ para cada semi-aresta $(u,v) \in D(f)$.

Reciprocamente, a face externa h tem forma retangular se:

- $\alpha(u,v) \geq 2$
- $\beta(u,v) = 0$ para cada semi-aresta $(u,v) \in D(h)$.

O algoritmo *Tidy-Rectangle-Compact* que utiliza o modelo de fluxo de rede no problema de *compactação* é mostrado a seguir. Ou seja, dois fluxos de rede são construídos, um para segmentos horizontais e outro para segmentos verticais, N_{hor} (*Figura 2.25*) e N_{ver} (*Figura 2.26*).

Algoritmo Tidy-Rectangle-Compact

Entrada: Embutimento planar do grafo *G* com *n* vértices de grau máximo 4;
representação ortogonal *G*, tal que todas as faces tenham forma retangular.

Saída: Desenho ortogonal planar no *grid* ψ de *G* com representação ***H***, de forma que a altura, largura, área e tamanho total das arestas sejam mínimos.

1. Constrói o fluxo de rede N_{hor} e N_{ver} associados a ***H***.
2. Calcula o fluxo de custo mínimo para N_{hor} e N_{ver}.
3. O conjunto de *tamanhos* de cada segmento de ***H*** é igual ao fluxo no arco correspondente em N_{hor} e N_{ver}.

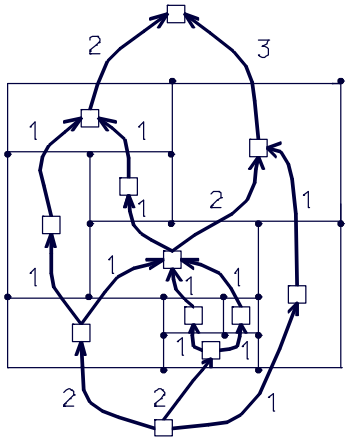


Figura 2.25 - Rede N_{hor} para a representação ortogonal da Figura 2.20, e o fluxo de custo mínimo para N_{hor}.

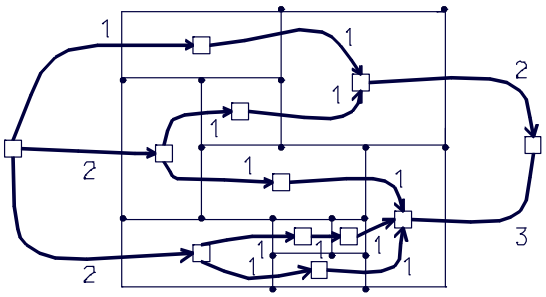


Figura 2.26 - Rede N_{ver} para a representação ortogonal da Figura 2.20, e o fluxo de custo mínimo para N_{ver}.

A rede N_{hor} tem um nó associado a cada face interna, mais dois nós especiais, denotados por *s* e *t*, representando a região inferior (*lower*) e a região superior (*upper*) da face externa, respectivamente. N_{hor} tem também um arco (*f*,*g*) para todo par de faces *f* e *g* que compartilha um segmento horizontal *e*, com *f* abaixo de *g*. O fluxo no arco (*f*,*g*) representa o tamanho do segmento *e*. Consequentemente, o arco (*f*,*g*) tem limite inferior $\lambda(f,g) = 1$, capacidade $\mu(f,g) = +\infty$ e custo $\chi(f,g) = 1$. As *Figuras 2.25 e 2.26* mostram um fluxo de custo mínimo para N_{hor} e N_{ver}, respectivamente.

As seguintes propriedades da rede N_{hor} são imediatas:

- N_{hor} é planar e acíclica, com um único nó fonte e nó destino, ambos na face externa. Isto implica que o N_{hor} é um grafo *s*-*t* planar.
- N_{hor} tem O(*n*) nós e arcos.

O mesmo raciocínio segue para N_{ver}.

Além disso:

- Os valores dos fluxos de N_{hor}(N_{ver}) são iguais à largura (altura) do desenho.
- A soma dos custos do fluxo em N_{hor} e N_{ver} é igual a soma total dos tamanhos das arestas verticais e horizontais no desenho.

O problema da *compactação*, para uma representação ortogonal ***H***, com faces retangulares, pode ser reduzido à computação do problema de fluxo de custo mínimo e é obtido pelo algoritmo *Tidy-Rectangle-Compact*. A complexidade (tempo) dos passos 1 e 3 é O(n) e do passo 2 é O($n^{7/4}$ log n), onde n representa o número de vértices.

2.4.11. Método de compactação unidimensional

Devido a existência de instâncias de tamanho muito grandes, as pesquisas em desenho de circuitos integrados (VLSI) focaram em métodos unidimensionais. Somente uma dimensão pode ser tratada a cada vez, sendo que a outra dimensão se mantém fixa. O problema de *compactação* unidimensional será referenciado como sendo $COMP^1_{soma}$, $COMP^1_{área}$ e $COMP^1_{max}$.

Após a etapa de *compactação*, o *layout* é modificado, alternando a direção e executando um outro passo, o que resulta em um processo iterativo. Além disto, a cada passo, as decisões são puramente locais e a *compactação* em uma direção pode impedir maior progresso em outras direções. Consequentemente, o *layout* pode ser bloqueado em ambas as dimensões, mas ainda estar longe de uma solução ótima. Veja *Figura 2.27* para ilustração.

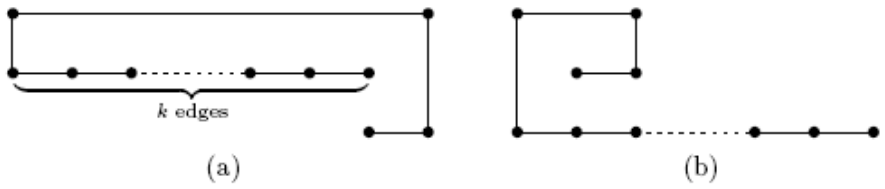


Figura 2.27 - (a) ambas as direções são bloqueadas, o tamanho total das arestas é 2k + 5. (b) um layout obtido por um método ótimo de compactação com tamanho total de arestas k + 6 [25].

O método *compression-ridge*, originário de projetos VLSI, procura por cortes que dividam o *layout* em duas partes e que passam através de regiões de espaço vazio. Para um embutimento fixo, o espaço vazio corresponde às arestas que são maiores que o tamanho mínimo de uma unidade. Se tal corte é encontrado, suas arestas podem ser diminuídas de pelo menos uma unidade e o embutimento no *grid* resultante é ainda possível.

Será esboçado o método relacionado com o *fluxo máximo em rede* [22], adaptado para diferentes cenários na área de desenho de grafos. Todos os cortes são vistos como uma interpretação de um fluxo máximo em uma rede *N* que depende do desenho inicial. O passo de *compactação* na direção de *x* é mostrado na *Figura 2.27*. A *compactação* na direção de *y* pode ser pensada de maneira similar. Primeiramente, o *layout* é cortado em faixas horizontais. Isto corresponde ao processo de corte, que é detalhado em [22], com a restrição de que somente arestas artificiais na direção horizontal são permitidas. A modificação no método do corte ainda gasta tempo de execução linear e o resultado é um desenho com faces internas de forma retangular. Agora a rede *N* pode ser construída como segue, sendo cada face retangular *f* correspondente a um nó *n*(*f*) em *N*. Além disto, existem dois nós *s* e *t* para a face externa, *s* no topo e *t* na parte de baixo do desenho. Os arcos são direcionados de cima para baixo. Para cada aresta horizontal *e*, separando uma face superior *f* de uma face inferior *g*, existe um arco $a^{+}_{e} = (n(f), n(g))$ e um arco $a^{-}_{e} = (n(g), n(f))$. A capacidade de a^{+}_{e} é o tamanho de *e* menos 1. Isto corresponde à maior possibilidade de "encurtamento" de *e*. A capacidade de ∞ é atribuído ao arco oposto a^{-}_{e}, levando em consideração possíveis alongamentos de *e*. O fluxo máximo de *s* para *t* na rede corresponde a maior redução que se pode aplicar para se obter um desenho de largura mínima. Assim, obtém-se um *layout* de largura horizontal ótima. Cada passo da *compactação* gasta tempo O (*n logn*) em número de vértices, o gargalo é a computação do problema do fluxo máximo em *N*. Por construção, a rede *N* é planar e linear no tamanho (número de vértices) do grafo original.

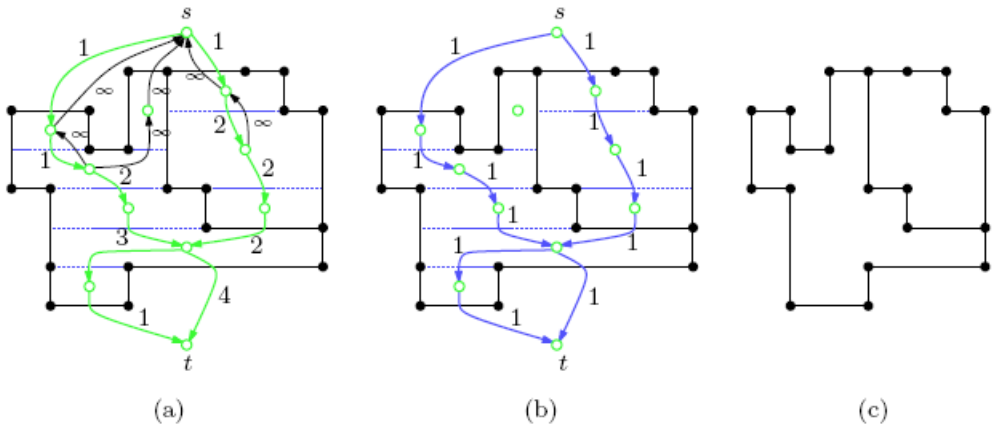


Figura 2.28 - Método compression-ridge em desenho de grafos: (a) a rede N para a compactação x (somente alguns dos arcos para cima são mostrados). (b) o fluxo máximo em N; (c) o desenho após o passo de compactação [22].

O chamado *método de compactação baseado em grafos* [18] será discutido nesta seção. O mesmo representa uma abordagem diferente e mais eficiente (de acordo com análise feita em [22]). Nesta abordagem, tem-se dois grafos de *layout*, sendo um para cada direção (*x* ou y) da *compactação*, ou seja, *compactação* horizontal e vertical, o que codifica as

propriedades da visibilidade (define quais vértices estão à direita, acima, à esquerda e abaixo de outros vértices) entre os caminhos maximalmente conectados, vertical e horizontalmente, de um dado embutimento no *grid*. Estes caminhos são também chamados de barras em [18] e de segmentos em [24], [25] , [26].

Definição – Um segmento horizontal é um componente maximalmente conectado em (V,E_h), com o subgrafo de G contendo somente arestas horizontais. Similarmente, são definidos os segmentos verticais em (V,E_v). Os conjuntos S_h e S_v referem-se aos segmentos horizontais e verticais, respectivamente, e o conjunto $S = S_h \cup S_v$ refere-se ao conjunto de todos os segmentos verticais e horizontais. Um vértice v está sobre dois únicos segmentos, $hor(v) \in S_h$ e $ver(v) \in Sv$.

Os grafos direcionados de *layout* $D_x = (V_x, A_x)$ e $D_y = (V_y, A_y)$ também conhecidos como D_h e D_v são construídos da seguinte forma: o conjunto de nós V_x do grafo horizontal D_x corresponde ao conjunto de segmentos verticais S_v. Uma construção similar se aplica para D_y , onde $V_y = S_h$.

Para um conjunto de arcos A, seja $trans(A)$ o fecho transitivo de A. As relações geométricas entre os segmentos definem o conjunto de arcos nos dígrafos: Sempre que um segmento horizontal S_i está a esquerda de um outro segmento horizontal S_j, deseja-se encontrar um caminho direcionado entre S_i e S_j. As relações verticais são caracterizadas analogamente. Formalmente, tem-se:

$trans(A_x) = \{ (S_i, S_j) \mid S_i$ está a esquerda de $S_j \}$ e
$trans(A_y) = \{ (S_i, S_j) \mid S_i$ está abaixo de $S_j \}$.

Quaisquer conjuntos que tenham as propriedades acima podem ser usados como conjuntos de arcos do grafo de *layout*. A *Figura 2.29* (retirada de [22] e modificada pelo acréscimo de alguns arcos que garantem a restrição de distância entre os segmentos) mostra grafos de *layout* para o exemplo citado nesta seção com os conjuntos de arcos produzidos por um método de varredura por retas [88].

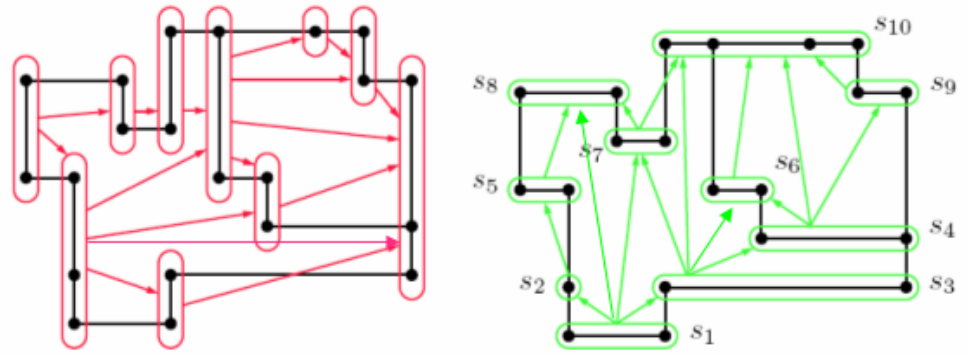


Figura 2.29 - Os grafos direcionados de layout D_x (esquerda) e D_y (direita) [22], com base no grafo da Figura 2.28.

Cada arco corresponde a uma restrição de distância para um par de segmentos. Desde que as propriedades de visibilidade sejam mantidas na *compactação* uni-dimensional (relembrando que as coordenadas na outra direção são fixas), um arco (S_i, S_j) descreve o fato que a todos os vértices de S_j devem ser atribuídos uma maior coordenada que aquelas para os vértices de S_i. Consequentemente, as tarefas da *compactação* uni-dimensional na direção de *x* se reduzem à computação da numeração topológica para os nós em D_x. Similarmente, o passo da *compactação* vertical corresponde à numeração topológica em D_y.

2.4.12. Método de compactação bidimensional

O problema de *compactação* bidimensional para um desenho ortogonal no *grid* ψ está relacionado com a tarefa de alterar as coordenadas dos vértices e segmentos de arestas, enquanto preserva-se a forma do desenho. Desta maneira o tamanho total das arestas é minimizado. Este problema tem uma relação muito aproximada com o problema de *compactação* bi-dimensional em desenho de circuitos integrados (VLSI) que é NP-difícil [15].

Para caracterizar o conjunto de soluções possíveis para o problema de *compactação* bi-dimensional, em termos de *caminho* Mutzel *et al*, [24] e [25] introduziram conceitos importantes, os quais permitiram definir e formular o problema de forma a utilizar os conceitos de programação linear inteira (PLI).

A idéia era atacar o problema com base na abordagem estabelecida por [50], que dizia o seguinte: "*A dificuldade da compactação bidimensional está em determinar como as duas dimensões do layout devem interagir para minimizar a área do desenho"*.

A metodologia apresentada por Mutzel *et al,* [24] e [25], objetiva atacar exatamente este ponto. Ela provê condições necessárias e suficientes para todas as soluções possíveis de uma dada instância do problema de *compactação*. Estas condições são baseadas na existência de um *caminho* nos chamados *grafos restritos* em relação às direções *x* e *y*.

Esta idéia é descrita da seguinte forma: Os dois *grafos restritos* D_h e D_v especificam a *forma* de um dado desenho ortogonal. São caracterizadas exatamente estas extensões dos *grafos restritos*, os quais pertencem aos possíveis desenhos ortogonais no *grid*. A tarefa é estender os tais *grafos restritos* para um par de *grafos restritos completos* pela adição de arcos, de maneira que as condições necessárias e suficientes sejam satisfeitas e o tamanho total das arestas do *layout* seja minimizado. Consequentemente, o problema geométrico é transformado em um problema teórico em grafos. Além disto, pode-se construir estas instâncias, tendo somente uma *extensão completa*. Para estes casos, pode-se resolver o problema de *compactação* em tempo polinomial.

Assim, o problema resultante foi formulado como um problema de programação linear inteira (PLI) que pode ser resolvido através de métodos *branch-and-bound* ou *branch-and-cut*. Em Mutzel *et al* [24], [25] foi mostrado que é possível resolver o problema de *compactação* bi-dimensional para as 11.582 instâncias de grafos, testadas em tempo computacional curto. Além disto, os desenhos resultantes mostraram ser desenhos otimamente compactados. O tamanho total das arestas foi reduzido em 37%, quando comparado a técnicas unidimensionais, e em 65,4%, quando comparado ao método de *compactação* proposto por [17].

a) Caracterização de soluções possíveis

A transformação do problema geométrico de *compactação* em um problema teórico em grafos será descrita após as definições básicas mostradas. A noção de *descrição de forma* será introduzida e algumas de suas propriedades serão apresentadas. A correspondência *um-para-um* entre a *descrição de forma* (satisfazendo uma certa propriedade) e possíveis desenhos ortogonais no *grid* será estabelecida.

b) Definições e notações

Em um desenho ortogonal no *grid* ψ de um grafo *G*, os vértices são colocados em pontos com coordenadas inteiras mutuamente distintos no *grid* e as arestas em *caminhos* (linhas no *grid* mutuamente distintas), conectando seus pontos extremos. Denomina-se ψ como *simples* se o número de dobras e cruzamentos em ψ são iguais a zero. Todo desenho ortogonal no *grid* pode ser transformado em um desenho *simples*. A transformação direta

consiste da substituição dos cruzamentos e dobras por vértices virtuais conforme, *Figura 2.30*.

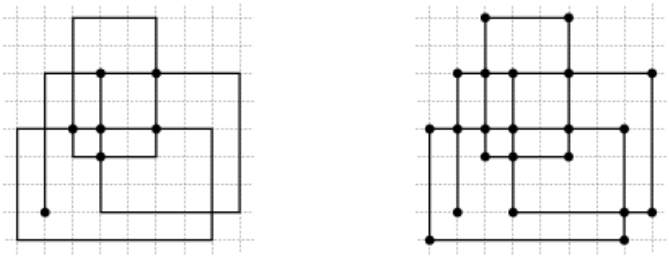


Figura 2.30 - Um desenho ortogonal no grid e sua transformação para desenho simples [25].

A forma do desenho *simples* é dada pelos ângulos internos às faces. Pode ser observado que a noção de forma induz ao particionamento do desenho em classes equivalentes. Frequentemente, a forma de um desenho ortogonal é dada pela chamada *representação ortogonal* ***H***. Formalmente, para um desenho ortogonal *simples*, ***H*** é uma função de um conjunto de faces *F* para uma lista ordenada no sentido horário, de tuplas (e_r, a_r) onde e_r é uma aresta e a_r o ângulo formado com a próxima aresta dentro da face apropriada. Usando as definições, pode-se então especificar o problema de *compactação*:

Definição 1: *O problema de compactação para um desenho ortogonal (do inglês, COD) é formalmente dado pelo seguinte: Dado um desenho ortogonal no grid* ψ, *com representação ortogonal* ***H***, *encontre um desenho* ψ *' com representação ortogonal* ***H***, na *qual o tamanho total das arestas seja mínimo.*

A *Figura 2.27 (a)* mostra um desenho ortogonal com tamanho total das arestas igual a 2*k* + 5, o qual não pode ser melhorado por um método de *compactação* unidimensional. A razão disto está relacionada com o fato que um método de *compactação* unidimensional é baseado nas propriedades da abordagem de visibilidade. Um método de *compactação* bi-dimensional resulta no desenho da *Figura 2.27 (b*), cujo tamanho total das arestas é somente *k* + 6.

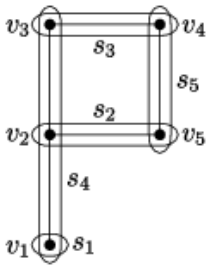
Seja ψ um desenho ortogonal no *grid* de um grafo *G* = (*V*,*E*). Ele induz a partição de um conjunto de arestas *E* em conjuntos de arestas horizontais E_h e verticais E_v. Um conjunto de sub-segmentos δ horizontal (vertical) em ψ é um componente conectado em (*V*, E_h) ((*V*, E_v)). Se o componente é maximalmente conectado ele é também denominado como um segmento que faz parte de um conjunto denominado por *S*. Denota-se o conjunto de sub-segmentos horizontais e verticais por δ_h e δ_v, respectivamente, e os conjuntos de segmentos por S_h e S_v, de maneira que $S_h \subseteq \delta_h$, $S_v \subseteq \delta_v$. Define-se $\delta = \delta_h \cup \delta_v$ e $S = S_h \cup S_v$. As seguintes observações são de interesse (ver *Figura 2.31*):

1. Cada aresta é um sub-segmento, por exemplo, $E_h \subseteq \delta_h$, $E_v \subseteq \delta_v$.
2. Cada vértice v pertence a um segmento horizontal (hor(v)) e a um segmento vertical (vert(v)) simultaneamente.
3. Cada sub-segmento s está contido em exatamente um segmento, chamado de seg(s).
4. Sejam v_l, v_r,v_b *e* v_t os vértices em s, mais a esquerda, mais a direita, mais inferior e mais superior, respectivamente. Então, os limites de um sub-segmento são dados por : $l(s)$ = vert(v_l), $r(s)$ = vert(v_r), $b(s)$ = hor(v_b) *e* $t(s)$ = hor(v_t).

O seguinte **Lema** implica que o número total de segmentos é 2|V| - |E|.

Lema 2.3. *Seja* ψ *um desenho ortogonal simples no grid, de um grafo* $G = (V, E_h \cup E_v)$. *Então o* $|S_h| = |V| - |E_h|$ *e* $|S_v| = |V| - |E_v|$.

A prova para este lema pode ser vista em [24].



$S_h = \{s_1, s_2, s_3\}$ $S_v = \{s_4, s_5\}$

i	$V(s_i)$	$E(s_i)$	$l(s_i)$	$r(s_i)$	$b(s_i)$	$t(s_i)$
1	$\{v_1\}$	$\{\}$	s_4	s_4	s_1	s_1
2	$\{v_2, v_5\}$	$\{(v_2, v_5)\}$	s_4	s_5	s_2	s_2
3	$\{v_3, v_4\}$	$\{(v_3, v_4)\}$	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_1	s_3
5	$\{v_4, v_5\}$	$\{(v_4, v_5)\}$	s_5	s_5	s_2	s_3

Figura 2.31 - Segmentos de um desenho ortogonal simples no grid e seus limites [25].

c) Descrição de forma e suas propriedades

Seja $G = (V,E)$ um grafo com uma representação ortogonal $\boldsymbol{H}$ simples e segmentos $S_h \cup S_v$. Uma *descrição de forma* de $\boldsymbol{H}$ é uma tupla $\sigma = \langle (D_h, D_v) \rangle$ dos *grafos restritos*. Ambos grafos são direcionados e definidos como $D_h = (S_v, A_h)$ e $D_v = (S_h, A_v)$. Assim, cada nó em D_h e D_v é um segmento e este conjunto de segmentos são unidos por arcos em A_v *e* A_h,. Dessa forma, tem-se que:

$$A_h = \{(l(e),r(e)) \mid e \in E_h\} \text{ e } A_v = \{(b(e),t(e)) \mid e \in E_v\} \quad (2.13)$$

Os dois dígrafos caracterizam um relacionamento conhecido entre os segmentos que devem ser verificados por qualquer desenho de um grafo devido às suas propriedades de forma. Seja $a = (s_i,s_j)$ um arco $A_h \cup A_v$. Se $a \in A_v$ então o segmento horizontal s_i deve ser colocado em pelo menos uma unidade de *grid* abaixo do segmento s_j.

Para os segmentos verticais o arco $a \in A_h$, ou seja, expressa o fato que s_i deve estar à esquerda de s_j. Cada arco é definido por pelo menos uma aresta em E. Claramente, cada aresta vertical determina a posição relativa de dois segmentos horizontais e vice-versa. A *Figura 2.32* ilustra a descrição de forma de um desenho ortogonal simples no *grid*.

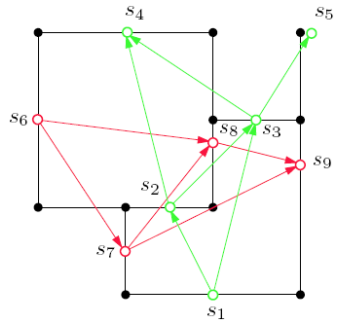


Figura 2.32 - Um desenho ortogonal simples e sua descrição de forma. [22].

De acordo com a *Figura 2.32* tem-se que:

Segmentos horizontais $S_h = \{s_1, s_2, s_3, s_4, s_5\}$
Segmentos verticais $S_v = \{ s_6, s_7, s_8, s_9\}$
Arcos horizontais $A_h = \{(s_6, s_8), (s_6, s_7), (s_7, s_8), (s_7, s_9), (s_8, s_9)\}$
Arcos verticais $A_v = \{(s_1, s_2), (s_1, s_3), (s_2, s_4), (s_2, s_3), (s_3, s_4), (s_2, s_3), (s_3, s_4), (s_3, s_5)\}$
Grafo restrito $D_h = \{S_v, A_h\}$
Grafo restrito $D_v = \{S_h, A_v\}$.

Para dois vértices, v e w, é utilizada a notação $v \xrightarrow{*} w$ denotando a existência de um caminho direcionado de v para w. A *descrição de forma* tem as seguintes propriedades:
Lema 2.4. *Seja* $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ *uma descrição de forma. Para cada sub-segmento* $s \in \delta_h \cup \delta_v$, *os caminhos* $l(s) \xrightarrow{*} r(s)$, $b(s) \xrightarrow{*} t(s)$ *estão contidos em* $A_h \cup A_v$. A prova deste **Lema** pode ser vista em [24], [25].

Definição 2. Sejam os pares de segmentos $(s_i, s_j) \in \delta \times \delta$. Denomina-se o par de *separado*, se e somente se uma das seguintes condições é verificada:

1. $r(s_i) \xrightarrow{*} l(s_j)$ (s_i está a esquerda de s_j) (2.14)
2. $r(s_j) \xrightarrow{*} l(s_i)$ (s_i está a direita de s_j) (2.15)
3. $t(s_j) \xrightarrow{*} b(s_i)$ (s_i está acima de s_j) (2.16)
4. $t(s_i) \xrightarrow{*} b(s_j)$ (s_i está abaixo de s_j) (2.17)

Em um desenho ortogonal, pelo menos uma das quatro condições deve ser satisfeita para qualquer par (s_i, s_j). As duas observações seguintes mostram que somente é necessário considerar segmentos *separados* de direções opostas.

Observação 1. Seja (s_i, s_j) $\in$ δ x δ um par de sub-segmentos. Se (*seg*(s_i), *seg*(s_j)) é *separado* então (s_i, s_j) é *separado*.

Observação 2. Assume-se que os arcos entre os segmentos formam um grafo acíclico. A seguinte afirmativa é verdade: Todos os pares de segmentos são *separados* se e somente se todos os pares de segmentos de direções opostas são *separados*.

O seguinte **Lema** mostra que se pode restringir o foco para segmentos separados que compartilhem uma face em comum. Para uma face *f* pode-se escrever *S(f)* para os segmentos contendo as arestas horizontais e verticais no contorno de *f*.

Lema 2.5. *Todos os pares de segmentos são separados se e somente se para cada face f os pares de segmentos* (s_i, s_j) $\in$ *S(f)* x *S(f) são separados.* A prova para este **Lema** pode ser vista em [24].

d) Extensões completas da descrição de forma

Qualquer *descrição de forma* $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ pode ser extendida de maneira que os *grafos restritos* resultantes correspondam a um possível desenho ortogonal planar. Caracterizam-se estas *extensões completas* em termos de propriedades de seus *grafos restritos*.

Definição 3. Uma extensão completa de uma *descrição de forma* $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ 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. Todos os pares de segmentos são *separados*.

O seguinte **Teorema** caracteriza o conjunto de soluções possíveis para o problema de *compactação*.

Teorema 2.2. *Para qualquer desenho ortogonal simples com descrição de forma* $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$, *existe uma extensão completa* $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ *de* σ *e vice versa, sendo que qualquer extensão completa* $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ *de uma*

descrição de forma $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ *corresponde a um desenho ortogonal simples, com descrição de forma* σ. A prova bem detalhada para este **Teorema** pode ser vista em [24], [25].

Prova – Para provar a primeira parte do **Teorema**, considera-se um desenho ortogonal simples no *grid* ψ com descrição de forma $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$. Seja $c(s_i)$ com $c \in N$, uma coordenada para o segmento $s_i \in S_h \cup S_v$. A extensão completa $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ é construída para σ como segue: $B_h = \{(s_i, s_j) \in S_v \text{ x } S_h \mid c(s_i) < c(s_j)\}$. Por exemplo, um arco é inserido a partir de cada segmento vertical a esquerda até cada segmento vertical à direita de s_i. De maneira similar, B_v é construída. Assim, tem-se $A_h \subseteq B_h$ *e* $A_v \subseteq B_v$. Mostra-se a completude por contradição, assumindo-se primeiro que existe algum par (s_i, s_j) que não seja separado. De acordo com a construção, isto somente é possível se os segmentos se cruzam em ψ, o que é uma contradição. Agora é assumido que existe um ciclo em um dos conjuntos de arcos. Novamente, a construção de B_h e B_v impedem este caso. Consequentemente τ é uma extensão completa de σ.

Será dada a prova construtiva para a segunda parte do **Teorema** pela especificação de um desenho ortogonal simples no *grid* para a extensão completa τ. Para completar esta tarefa, é necessário atribuir tamanhos aos segmentos. Uma atribuição de tamanho para uma extensão completa de uma descrição de forma $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ é uma função c: $S_h \cup S_v \rightarrow N$, com a propriedade $(s_i, s_j) \in B_h \cup B_v \Rightarrow c(s_i) < c(s_j)$. Dado τ, tal função pode ser calculada usando algum algoritmo de ordenação topológica em grafos acíclicos em τ. Por exemplo, caminho mais longo ou algoritmos de fluxo máximo no grafo dual. Para a atribuição de um tamanho fixo, o seguinte método, simples e direto, atribui as coordenadas para os vértices do grafo em análise. Seja $x \in N^V$ *e* $y \in N^V$ o vetor de coordenadas. Então, simplesmente seta-se $x_v = c(\text{vert}(v))$ e $y_v = c(\text{hor}(v))$ para cada vértice $v \in V$ e isto resulta em um desenho correto no *grid*. Os seguintes pontos tem que ser verificados:

1. Todas as arestas possuem tamanhos inteiros e positivos. O tamanho de uma aresta horizontal $e \in E_h$ é dado por $c(r(e)) - c(l(e))$. Sabe-se que ambos valores são inteiros e de acordo com o **Lema 2.4**, que $(l(e), r(e)) \in B_h$ e então $c(r(e)) > c(l(e))$. Um argumento similar se aplica para arestas verticais.

2. O desenho ortogonal ψ mapeia cada circuito do grafo *G* em um polígono retilíneo. Dada uma face *f*, $\psi(f)$ é uma representação geométrica de vértices e arestas pertencentes a face *f*. Isto é suficiente para mostrar que aquele $\psi(f)$ é um polígono retilíneo para cada face *f* em *G*. Cada vértice *v* no contorno de *f* é colocado de acordo com os segmentos hor(*v*) e ver(*v*). Dois vértices consecutivos *v* e *w* no contorno de *f* também compartilham o mesmo segmento horizontal ou vertical (desde que eles estejam ligados por uma aresta). Assim, $x_v = x_w$ ou $y_v = y_w$.

3. Sem cruzamento de subsegmentos. Caso contrário, é assumido que existem dois desses segmentos s_i e s_j que se cruzam. Então $c(r(s_i)) \geq c(l(s_j))$, $c(r(s_j)) \geq c(l(s_i))$, $c(t(s_j)) \geq c(b(s_i)$ e $c(t(s_i) \geq c(b(s_j))$ (veja também a *Figura 2.32*). Esta é uma contradição para a completude de τ.

É importante lembrar que uma única aresta ou um conjunto de arestas podem representar um segmento.

O problema de *compactação* é transformado em um problema *teórico em grafo*. A nova tarefa é encontrar uma *extensão completa* da *descrição de forma* σ dada, que minimize o tamanho total das arestas. Se a *descrição de forma* já satisfaz a condição de uma *extensão completa* (veja *Figura 2.34*(*a*)), o problema da *compactação* pode ser resolvido otimamente em tempo polinominal. Dessa forma, o problema resultante é o *dual* do problema de fluxo máximo. Algumas vezes, a *descrição de forma* não é completa, mas é somente possível estendê-la em uma direção (veja *Figura 2.34(b*)). Nesses casos também é fácil resolver o problema de *compactação*. Mas na maioria dos casos não é claro o modo como estender a descrição de forma, visto que existem muitas possibilidades diferentes (veja *Figura 2.34(c)*).

e) Formulação PLI para o problema de compactação

A caracterização dada na seção anterior pode ser usada para obter uma formulação de programação linear inteira para o problema da *compactação* COD. Seja ψ um desenho ortogonal no *grid* simples de um grafo $G = (V, E_h \cup E_v)$, com a representação ortogonal $\boldsymbol{H}$, e seja $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ a *descrição de forma* correspondente. O conjunto de soluções possíveis do COD pode agora ser escrito como $\varsigma(\sigma) = \{\tau \mid \tau$ é a *extensão completa* de $\sigma\}$. Seja um $C = S_h \times S_h \cup S_v \times S_v$ o conjunto de arcos possíveis nos dígrafos (D_h e D_v) e σ. $\mathcal{Q}^{|C|}$ é o vetor no espaço cujos elementos são indexados por números correspondendo aos elementos de *C*. Para uma *extensão completa* $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ de σ define-se um elemento $x^\tau \in \mathcal{Q}^{|C|}$ da seguinte maneira: $x_{ij}{}^\tau = 1$ se $(s_i, s_j) \in B_h \cup B_v$ e $x_{ij}{}^\tau = 0$, caso contrário. Utiliza-se estes vetores para caracterizar a *compactação* do politopo $P_{COD} = \text{conv}\{ x^\tau \in \mathcal{Q}^{|C|} \mid \tau \in \varsigma(\sigma)\}$.

A *Figura 2.33* ilustra os cruzamentos de subsegmentos em direções iguais e opostas. A *Figura 2.34* ilustra a descrição de forma.

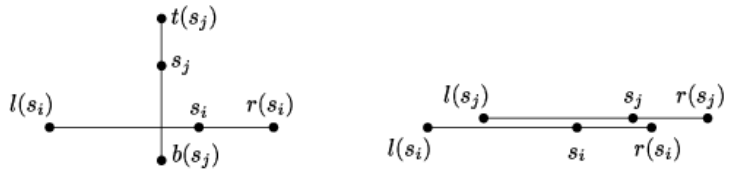


Figura 2.33 - Cruzamento de sub-segmentos em direções opostas (esquerda) e iguais (direita) [25].

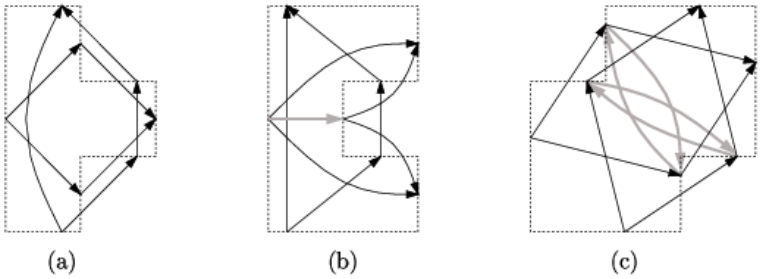


Figura 2.34 - Três tipos de descrição de forma. Linhas pontilhadas mostram o desenho ortogonal no grid, arcos como setas finas na descrição de forma e arcos cinzas grossos são completudes possíveis [25].

Para determinar o mínimo do tamanho total das arestas sobre todos os pontos possíveis em P_{COD}, introduz-se um vetor $c \in \mathcal{Q}^{|S_h \cup S_v|}$ para codificar a atribuição do tamanho e dar uma formulação de programação linear inteira para o problema de *compactação* COD. Seja *M* um número muito grande (para a escolha de *M* veja o **Lema 2.7**). A PLI para o problema de *compactação* é a seguinte:

$$\min \sum_{e \in E_h} c_{r(e)} - c_{l(e)} + \sum_{e \in E_v} c_{t(e)} - c_{b(e)} \tag{2.18}$$

Sujeito a:

$$x_{ij} = 1 \qquad \forall (s_i, s_j) \in A_h \cup A_v \tag{2.19}$$

$$x_{r(i),\,l(j)} + x_{r(j),\,l(i)} + x_{t(j),\,b(i)} + x_{t(i),\,b(j)} \geq 1 \qquad \forall (s_i, s_j) \in S \times S \tag{2.20}$$

$$c_i - c_j + (M+1)x_{ij} \leq M \qquad \forall (s_i, s_j) \in C \tag{2.21}$$

$$x_{ij} \in \{0,1\} \qquad \forall (s_i, s_j) \in C \tag{2.22}$$

Na *Equação* (*2.18*) o identificador de aresta *e* pode ser substituído também pelo identificador do segmento *s* quando este engloba mais de uma aresta.

A *Equação* (*2.18*) representa a soma total dos comprimentos das arestas. A *Equação* (*2.19*) está relacionada com a existência do arco na extensão completa da descrição de forma. A *Equação* (*2.20*) indica q existência do caminho entre duas arestas verticais ou horizontais.

A *Equação* (*2.21*) enfatiza a garantia de que qualquer outra aresta seja menor do que M e finalmente a *Equação* (*2.22*) indica que o arco existe ou não na extensão completa da descrição de forma.

A função objetivo expressa a soma dos tamanhos de todas as arestas no desenho de *G*. Observe que a formulação também captura o problema relacionado à *minimização do tamanho da aresta mais longa* em um desenho. Neste caso, a restrição $c_{r(e)} - c_{l(e)} \leq l_{max}$ ou o $c_{t(e)} - c_{b(e)} \leq l_{max}$ deve ser adicionada para cada aresta *e* e a função objetivo deve ser substituída pelo mínimo l_{max}. Além disso, é possível dar a cada aresta um peso individual na função objetivo. Desta maneira, arestas com valores mais altos são consideradas mais importantes e a elas serão, preferencialmente, atribuídos tamanhos menores.

Será mostrada a motivação informal dos três diferentes tipos de restrições de desenho e, então, será mostrado que qualquer solução possível de uma formação PLI, de fato, corresponde a um desenho ortogonal no *grid*.

1. *Restrições de forma*: Procura-se por uma extensão de uma *descrição de forma* σ. Desde que qualquer extensão da descrição de forma contenha um conjunto de arcos de σ, devem ser atribuídas o valor 1 às entradas apropriadas de *x*.
2. *Restrições de Completude:* Esse conjunto de restrições garante a completude da extensão da descrição de forma. As respectivas inequações modelam as *restrições de consistência* abaixo.
3. *Restrições de Consistência:* O vetor *c* corresponde aos tamanhos a serem atribuídos e assim se deve satisfazer a propriedade $(s_i, s_j) \in B_h \cup B_v \rightarrow c(s_i) < c(s_j)$. Se $x_{ij} = 1$, na inequação tem-se $c_j - c_i \geq 1$, satisfazendo a propriedade para o arco (i,j). Caso $x_{ij} = 0$, tem-se $c_i - c_j \leq M$, o qual é verdade se a *M* for atribuído o máximo entre largura e altura de ψ.

A observação seguinte, bem como o subsequente **Lema**, motiva o fato de que nenhuma restrição adicional, que proíba os ciclos, é necessária.

Observação 3. Seja (x,c_f) com $x \in \{0,1\}^{|C|}$ e $c_f \in \mathbb{Q}^{|S_h \cup S_v|}$ uma solução possível para PLI e seja z_f o valor de uma função objetivo. Então existe uma solução possível (x,c) para a PLI com $c \in \mathbb{N}^{|S_h \cup S_v|}$ e o valor da função objetivo $z \leq z_f$. A prova para esta observação pode ser vista em [25].

Lema 2.6. Seja *(x,c) uma solução possível para PLI e sejam* D_h *e* D_v *os dígrafos correspondentes a x. Então,* D_h *e* D_v *são acíclicos.* A prova deste **Lema** pode ser vista em [25].

Teorema 2.3. *Para cada solução possível (x,c) do PLI para uma descrição de forma* σ *existe um desenho ortogonal no grid simples* ψ *, cuja forma corresponde a* σ *e vice versa. O tamanho total das arestas de* ψ *é igual ao valor da função objetivo*.

A prova deste **Teorema** pode ser vista em [24].

Lema 2.7. *O valor* $max\{|S_h|, |S_v|\}$ *é uma escolha suficiente para o valor de M.*

Prova - Qualquer desenho ótimo de ψ tem largura $w \leq |S_v|$ e altura $h \leq |S_h|$: caso contrário, um passo de *compactação* unidimensional pode ser aplicado. *M* tem que ser grande o suficiente para "desabilitar" a restrição (2.21), se a entrada correspondente a *x* for zero. *M* deve estar no limite superior em distância para qualquer par de segmentos. Então a *M* deve ser atribuído o maior tamanho entre $|S_h|$ e $|S_v|$, de maneira a satisfazer os requisitos. Em termos mais simples, *M* deve assumir o máximo valor entre a largura e altura de ψ .

f) Heurística rápida para compactação ortogonal

A busca constante por melhores heurísticas que tornem o problema da *compactação* ortogonal, que é um problema da classe NP-difícil [15], mais rápida, levou a resultados interessantes que são mostrados em [26].

Em [26] é apresentado um novo algoritmo para *compactação* ortogonal que fornece desenhos ortogonais nos quais o tamanho dos vértices é dado como entrada do processo. Esta é uma restrição crítica para muitas aplicações práticas. Os algoritmos apresentados provêm melhoramentos nas heurísticas apresentadas por [24], [25] e oferecem uma complexidade de tempo linear em número de vértices para o pior caso, e de acordo com os autores, fornecendo também resultados interessantes na prática. Neste trabalho também é apresentada uma formulação PLI mais simplificada.

Assim, a formulação da PLI para a compactação é definida como:

$$\begin{aligned} &x_a - x_b \geq 1 \quad \forall (a,b) \in A_v \\ &y_a - y_b \geq 1 \quad \forall (a,b) \in A_h \\ &y_a \geq 0 \quad \forall s \in S_v \\ &x_a \geq 0 \quad \forall s \in S_h \end{aligned} \tag{2.23}$$

Nas equações acima deve-se minimizar as diferenças entre as coordenadas *x* e *y* para todos os arcos que representam o caminho entre as arestas horizontais e verticais.

Foi proposta a seguinte estratégia para solucionar o problema de *compactação* ortogonal através da heurística rápida.

1. Calcule a orientação e direção de G e $\boldsymbol{H}$.
2. Calcule a descrição de forma $\sigma = (D_h, D_v)$ ou $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$
3. Calcule a extensão completa τ de σ .
4. Resolva o problema correspondente ao caminho mais longo em D_h e D_v.
5. Atribua às coordenadas de acordo com as distâncias mais longas encontradas.

Será mostrado como se pode achar uma *extensão completa* τ. A heurística utilizada é baseada na técnica de decomposição retangular [17]. O ponto de início da estratégia de decomposição retangular é a observação de que se todas as faces dos grafos forem retangulares, pode-se facilmente solucionar o problema de *compactação* aplicando o caminho mais longo ou o algoritmo de fluxo máximo em rede. A ideia é de subdividir essas faces que não são retangulares em retângulos e então resolver o problema nesta subdivisão. Isto induz um embutimento válido no gráfico original. O que permanece é executar esta subdivisão eficientemente, o que pode ser feito procurando certos padrões de ângulos (a) na face. Denota-se ângulos de $\pi/2$ na face, com o valor '0', e $3\pi/2$, com o valor '1'. Sempre que o padrão 1 0 0 é encontrado, corta-se um retângulo da face e continua a busca por padrão na face remanescente. Veja a *Figura 2.35* para ilustração. Termina-se quando não houver mais padrões em nenhuma face. Utilizando uma lista (l) como estrutura de dados, a decomposição de retângulos pode ser feita em tempo linear.

O algoritmo *Inicia-lista*(face *f*) encontra uma lista com os padrões 1 0 0 em cada face do desenho.

Algoritmo *Inicia-lista*(face *f*)

Lista $l \leftarrow \phi$
Para *cada* (e = (v, w), a, l, d) em f **faça**
    // Seja d' a direção obtida rotacionando d em $\pi/2$;
    se a = 1 **então** anexe (0, *seg*(e), d) em l;
    se a = 3 **então** anexe (1, *seg*(e), d) em l;
    se a = 4 e $e \in E_v$ **então** anexe (1, *seg*(e), d), (1, *hor*(w), d') em l;
    se a = 4 e $e \in E_h$ **então** anexe (1, *seg*(e), d), (1, *ver*(w), d') para l;
Fim
Retorne ***l***

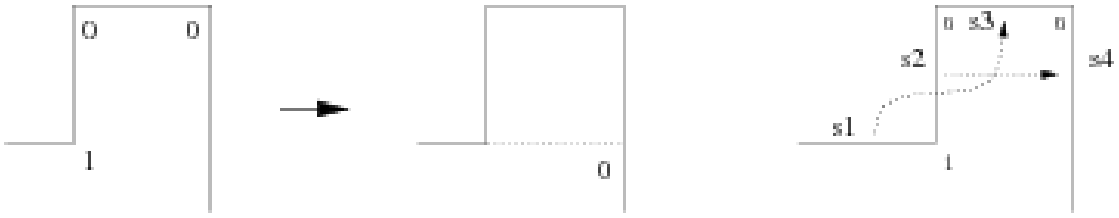


Figura 2.35 - Decomposição de uma face em um retângulo e a face remanescente [26].

A partir deste algoritmo pode-se diretamente derivar uma heurística para a completude. Considera-se, para isto, a situação ilustrada na *Figura 2.35*. Em vez de introduzir um nó fictício e uma aresta falsa no grafo, simplesmente acrescentam-se arestas ao grafo restrito.

No caso da *Figura 2.35* foi inserida a aresta (s_1, s_3) em D_v e a aresta (s_2, s_4) em D_h. Manuseia-se os outros três casos simetricamente. O algoritmo *define-box* descreve os quatro casos:

O algoritmo *Define-box (descrição da forma* σ, Direção d, Segmentos s_1, s_2, s_3, s_4) identifica os arcos necessários para compor cada face retangular do desenho. E o algoritmo *Decomponha* (*descrição da forma, Lista 1*) define as faces retangulares efetivamente.

Algoritmo *Define-box (descrição da forma* σ, Direção d, Segmentos s_1, s_2, s_3, s_4)

Seja $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$;
Se *d = cima* **então** $A_v \leftarrow A_v \cup (s_2, s_4)$, $A_h \leftarrow A_h \cup (s_3, s_1)$;
Se *d = baixo* **então** $A_v \leftarrow A_v \cup (s_4, s_2)$, $A_h \leftarrow A_h \cup (s_1, s_3)$;
Se *d = esquerda* **então** $A_v \leftarrow A_v \cup (s_1, s_3)$, $A_h \leftarrow A_h \cup (s_2, s_4)$;
Se *d = direita* **então** $A_v \leftarrow A_v \cup (s_3, s_1)$, $A_h \leftarrow A_h \cup (s_4, s_2)$;

Algoritmo *Decomponha* (*descrição da forma, Lista 1*)

Enquanto *tamanho* (*l*) > 4 **faça**
// denote com $t_i = (a_i, s_i, d_i)$ o *i-th* tupla em *l*;
se $(a_1 = 1)$ e $(a_2 = 0)$ e $(a_3 = 0)$ **então**
define-box (σ, d_1, s_1, s_2, s_3, s_4);
substitua t_1 com $(0, s_1, d_1)$;
remova t_2 e t_3 de *l*;

senão

mova t_1 para a parte de final de l;

fim

Fim

defina-box (σ, d_1, s_1, s_2, s_3, s_4).

O algoritmo de *completude* executa primeiro o algoritmo *inicia-lista* em toda face e em seguida executa o algoritmo *Decomponha* (*descrição da forma, Lista 1*).

Lema 2.8. *Com o algoritmo anterior, pode-se calcular uma extensão de forma completa de tamanho O(n) em tempo linear em número de vértices*.

Prova - Primeiramente, é mostrado que a *extensão completa* tem tamanho O(n). A *descrição de forma* inicial tem tamanho linear pela fórmula de *Euler*. Uma vez que a decomposição do retângulo introduz O(n) retângulos e são inseridas nela duas arestas por retângulo, a *extensão completa* tem tamanho linear. O tempo de execução linear decorre imediatamente deste fato também. Isto mostra que a extensão é completa. Seja um desenho G produzido pelo algoritmo convencional de decomposição de retângulo e seja um segmento vertical s_h e um segmento horizontal s_v que não sejam adjacentes. Devido ao desenho ser planar, s_h e s_v não se cruzam; assim, um dos quatro casos seguintes deve ser verificado:

a) s_h está acima de s_v;
b) s_h está abaixo de s_v;
c) s_h está a esquerda de s_v;
d) s_h está a direita de s_v.

Suponha-se que s_v esteja acima de s_h e que s_v não esteja à esquerda de s_h. Os outros casos são simétricos. Uma vez que s_v está acima de s_h, $s' = hor(\alpha(s_v))$ está também acima de s_h. Suponha que um dos casos seguintes seja verificado:

1. Existe um $s_u \in S_h$, o qual é interseção da projeção de s_u e s_v no eixo-y é não-vazia e existe um caminho nos arcos verticais para s_v em D_h.
2. Existe um segmento $s_w \in S_h$, o qual a interseção da projeção de s_w e s' no eixo-x é não-vazia e há um caminho de s_h para s_w em D_v.

Se a suposição for verdadeira, finaliza-se. Suponha que o segundo caso seja verificado. Então se traça uma linha paralela ao eixo-y com coordenadas x na interseção das

projeções. A partir dos retângulos que são interceptados pela linha paralela, pode-se construir facilmente um caminho de s_w para s' em D_v.
Foi dada uma prova construtiva que tanto s_u quanto s_w existem. O início se dá pelo segmento s_h e segue-se para o retângulo mais baixo a direita deste, se tal retângulo existir. Neste caso, segue-se deste retângulo para o retângulo mais à esquerda e acima. Itera-se até que um segmento seja encontrado, o qual induza a uma interseção das projeções. Devido ao fato de que ocorrem os crescimentos monotonicamente em coordenadas *x* e *y*, tal segmento deve existir por razões de monotonicidade. A existência do caminho é proveniente da maneira como se atravessa os retângulos.

2.5. Análise das metodologias discutidas neste capítulo

Neste capítulo foram discutidos conceitos e metodologias para desenho de grafos e as principais metodologias encontradas na literatura que são indicadas para o tratamento do problema de desenho de grafos ortogonais. Na abordagem *topologia-forma-métrica*, cada etapa possui seus algoritmos heurísticos, os quais foram discutidos em detalhe. Para as etapas de *ortogonalização* e *compactação* foram utilizadas também técnicas de programação linear inteira que mostram-se eficientes quando comparadas às heuristicas clássicas (*topologia-forma-métrica*). Cada heurística possui vantagens e desvantagens. Por exemplo, a heurística baseada na retangularização das faces do desenho e posterior utilização do modelo de fluxo de rede na solução do problema de *compactação* não garante que o desenho final seja otimizado. A abordagem que resolve o problema de *compactação* como um problema de PLI é interessante, uma vez que é possível obter modelos matemáticos que representam diversos critérios estéticos e resolvê-los de forma simples e eficiente. Porém, nesta abordagem, obter as funções objetivo de maneira automática não é trivial.

Com base no estudo deste capítulo, foi possível absorver o conhecimento necessário para o desenvolvimento de novas metodologias baseadas na *topologia-forma-métrica* e no algoritmo genético, como será visto nos próximos capítulos. Com isto, foi possível melhorar, significativamente, os resultados obtidos pela abordagem clássica *topologia-forma-métrica*.

CAPÍTULO 3

NOVAS METODOLOGIAS PARA DESENHO DE GRAFOS BASEADAS NA TOPOLOGIA-FORMA-MÉTRICA

No capítulo anterior, foram mostradas as dificuldades e limitações existentes em relação às abordagens clássicas para desenho automático de grafos ortogonais no *grid*. Neste capítulo, serão apresentadas e discutidas novas metodologias desenvolvidas para a solução de alguns dos problemas em aberto nesta área e seus resultados. Serão resolvidos os seguintes problemas: i) criação de novos modelos matemáticos para critérios estéticos que devem ser tratados na etapa de *compactação* da *topologia-forma-métrica*; solução do problema de compactação por técnicas de PLI; ii) solução do problema por PLI, em uma abordagem considerando simultaneamente os critérios modelados. Como são vários os critérios estéticos tratados, sendo que alguns deles conflitam entre si, tem-se um problema de otimização multiobjetivo. Este problema multiobjetivo é transformado em um problema mono-objetivo pela soma ponderada dos objetivos e resolvido por técnicas de PLI. Para aqueles critérios estéticos cujos modelos são não lineares, uma aproximação linear é aplicada [89], [90]; iii) solução do problema de fixar o embutimento planar na etapa de planarização da *topologia-forma-métrica*, utilizando o algoritmo genético na etapa de *planarização*, modelando o problema como um problema de otimização combinatória baseado em permutação de inteiros. Assim, o problema é resolvido em uma abordagem híbrida utilizando os algoritmos clássicos (*topologia-forma-métrica*) e o algoritmo genético (AG). Isto é feito de três maneiras distintas. A primeira delas considera a soma ponderada dos objetivos no processo evolucionário do AG. Já a segunda, considera a metodologia de tomada de decisão multicritério em ambiente *fuzzy* no processo evolucionário do AG e, finalmente, a terceira utiliza o algoritmo genético puramente multiobjetivo (NSGAII) na solução do problema de desenho automático de grafos ortogonais.

O desenvolvimento destas novas metodologias visa a apresentação de desenhos de grafos que atendam melhor e ao máximo de critérios estéticos possíveis, de acordo com a convenção de desenho ortogonal no *grid*. Além disto, permite flexibilidade para adição de critérios estéticos ou restrições de desenho que venham atender a necessidades diversas. A abordagem híbrida resolve o problema de fixar o embutimento planar na etapa de

planarização, possibilitando a obtenção de uma variedade de embutimentos que são tratados pelo algoritmo genético. Com estas metodologias foi possível obter desenhos mais otimizados ao final do processo.

As novas metodologias são descritas neste capítulo e seus resultados são apresentados e discutidos. Inicialmente serão modelados os critérios estéticos e aplicadas as técnicas de PLI na solução dos modelos criados.

3.1. Modelando os critérios estéticos na etapa de compactação

Para a convenção de desenho ortogonal no *grid*, na etapa de *compactação*, um número maior de critérios estéticos precisa ser tratado para garantir um desenho que respeite os requisitos de qualidade. Sendo assim, foram identificados os critérios estéticos que são relevantes nesta etapa e os seus modelos obtidos. São eles:

- *minimização da área do desenho*, que corresponde ao produto entre o maior e o menor lado do menor retângulo que contém todo o desenho;
- *minimização da razão de aspecto*, que corresponde à razão entre o maior e a menor lado do menor retângulo que contém todo o desenho;
- *minimização do tamanho da maior aresta do desenho.*

Lembra-se que o critério estético modelado em [24], [25] e [26] é a *minimização da soma total dos comprimentos das arestas* e deve se juntar aos demais critérios citados acima para a solução, de forma simultânea, do problema. Cada um destes critérios estéticos representa uma das funções objetivo que compõem o problema multiobjetivo na etapa de *compactação*. Os modelos obtidos são detalhados a seguir.

3.1.1. Minimização da área do desenho

A área do desenho de um grafo é um critério estético importante para o tratamento de desenhos ortogonais no *grid*. Esta área é obtida através do cálculo da área do menor retângulo que contém todo o desenho. Conforme citado no Capítulo 2, a habilidade para se construir desenhos com áreas eficientes é essencial em aplicações práticas, em que a economia de espaço em tela é de extrema importância. Isto ocorre, principalmente, quando a convenção de desenho no *grid* é adotada. Além disto, aplicações práticas podem conter grafos relativamente grandes.

O modelo matemático (PLI) que trata da *minimização da área do desenho* obtido neste estudo é mostrado a seguir:

$$\min\left((\max(c_{r(e)} - c_{l(e)}))*(\max(c_{t(e),} - c_{b(e)}))\right) \tag{3.1}$$

Este modelo é baseado na abordagem de segmentos horizontais e verticais, mostrada no capítulo 2. Nesta abordagem, $c_{r(e)}$ representa a coordenada *x* da extremidade direita da aresta *e*. $c_{l(e)}$ representa a coordenada *x* da extremidade esquerda da aresta *e*, $c_{t(e)}$ representa a coordenada *y* da extremidade superior da aresta *e* e $c_{b(e)}$ representa a coordenada *y* da extremidade inferior da aresta *e*. Quando calculamos, por exemplo, $c_{r(e)} - c_{l(e),}$ obtemos o tamanho da aresta *e*, que tem $c_{r(e)}$ como coordenada na extremidade direita e $c_{l(e)}$ como coordenada na extremidade esquerda de *e*. Logo, temos o tamanho da aresta *e*. O mesmo vale para os segmentos, apenas substituindo *e* por *s* na expressão (3.1).

Chamando de l_{hor} o maior segmento na horizontal e l_{ver} o maior segmento na vertical, tem-se $l_{hor} = \max(c_{r(e)} - c_{l(e)})$ e $l_{ver} = \max(c_{t(e),} - c_{b(e)})$. O modelo pode ser re-escrito como:

$$\min(l_{hor} * l_{ver}) . \tag{3.2}$$

Desta forma, tem-se o modelo matemático que representa o critério estético "*minimizar a área do desenho*" com base na abordagem dos segmentos horizontais e verticais como se deseja obter. Porém pode-se observar que este modelo é não-linear, sendo necessário utilizar técnicas de aproximação linear de modo a se obter uma representação linear do problema para tratá-lo como um problema de PLI.

Existem diversas técnicas para aproximação linear, como a aproximação linear utilizando o termo de primeira ordem da série de Taylor, o que representa a aproximação linear pela obtenção da derivada nas proximidades de um determinado ponto. Outra possibilidade é a aproximação linear utilizando uma abordagem paramétrica tratada dentro do contexto de "Fractional programming" [89], [90].

Na abordagem paramétrica, suponha que se tenham as funções $f_1(x)$ e $g_1(x)$ e a função objetivo não linear $min(f_1(x)\ x\ g_1(x))$. A aproximação linear para esta função pode ser feita por $min\ (f_1(x) + q\ g_1(x))$, onde $q \in \Re$ é uma espécie de parâmetro de ponderação. Esta aproximação linear é muito utilizada, devido a sua estrutura ser mais simples e melhor tratável computacionalmente. É importante enfatizar que se trata de uma aproximação linear, sendo assim, a solução encontrada não coincide com a solução exata, mas intuitivamente se aproxima da mesma [89]. Para os dois objetivos, "*minimização da área do desenho*" e "*minimização da razão de aspecto*", esta aproximação linear é utilizada neste trabalho. No caso da "*minimização da razão de aspecto*" onde a função objetivo não linear é $min(f_1(x) / g_1(x))$, a aproximação linear pode ser feita por $min\ f_1(x) - qg_1(x)$.

3.1.2. Minimização da Razão de aspecto

A razão de aspecto (*RA*) é a razão entre a maior e a menor aresta do menor retângulo que contém todo o desenho.

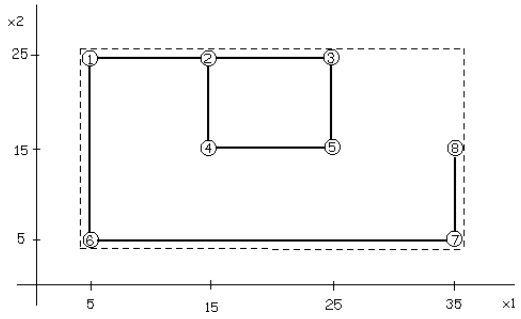


Figura 3. 1 – Exemplo de um grafo e o menor retângulo que contém todo o desenho.

De acordo com o exemplo da *Figura 3.1* a maior aresta do retângulo mede 30 unidades e a menor aresta do mesmo retângulo mede 20 unidades. Logo, a razão de aspecto (*RA*) para este desenho é dada por:

$$RA = \frac{30}{20} = 1{,}50 \tag{3.3}$$

Quanto mais próxima de um (1) é a razão de aspecto, melhor e mais bem distribuído em um determinado espaço será o desenho. Logo, pode-se deduzir que a razão de aspecto próxima de 1 proporcionará um desenho mais bem distribuído em uma tela de computador e, consequentemente, terá melhor legibilidade. Como a maioria das telas de computador mantém uma proporção pré-definida, como, por exemplo, 4/3, esta proporção poderá ser tratada como restrição no problema de programação linear inteira (PLI), para a função objetivo que trata a *minimização da razão de aspecto*. Embora a restrição de igualdade seja, na teoria, normalmente tratada em problemas de otimização, na prática foi observado que a imposição que a RA seja exatamente 4/3 levou a dificuldades na solução do problema de PLI, uma vez que se limitou muito o conjunto de soluções do problema. Este fato foi observado em testes executados para diversas instâncias dos grafos testados.

Outra abordagem poderia ser a de tratar a razão de aspecto não como uma função objetivo, mas como uma restrição da PLI que trata da *minimização da área do desenho*. Porém, na prática, esta restrição também levou a não-factibilidade do sistema de PLI.

Logo, o modelo foi obtido considerando-se a abordagem baseada na noção de *segmentos* e *descrição de forma*, tratada no Capítulo 2 (*Seção 2.4.12*), que traduz o desenho do grafo em modelos baseados em segmentos verticais e horizontais. Será necessário,

primeiramente, encontrar, no desenho, o maior entre todos os segmentos na vertical e o maior entre todos os segmentos na horizontal. Como visto na seção anterior, denomina-se por l_{hor} o maior segmento na horizontal e por l_{ver} o maior segmento na vertical. Obtém-se, assim, $l_{hor} = \max (c_{r(e)} - c_{l(e)})$ e $l_{ver} = \max (c_{t(e),} - c_{b(e)})$.

O modelo matemático para o objetivo "*minimização da razão de aspecto"* é dado por:

$$\min \ (\max(\max (c_{r(e)} - c_{l(e)}), \max (c_{t(e),} - c_{b(e)})) / (\min(\max (c_{r(e)} - c_{l(e)}), \max (c_{t(e),} - c_{b(e)}))). \quad (3.4)$$

Sujeito a:

$$\min(\max (c_{r(e)} - c_{l(e)}), \max (c_{t(e),} - c_{b(e)})) > 0.$$

E pode ser re-escrito como:

$$\min \ (\max(l_{hor}, l_{ver}) / (\min(l_{hor}, l_{ver}). \quad (3.5)$$

Sujeito a:

$$\min(l_{hor}, l_{ver}) > 0.$$

Para tratar o problema relacionado com a restrição que, caso se considere exatamente a proporção 4/3, por exemplo, leva a não-factibilidade do sistema, pode-se considerar apenas uma restrição que diz que $l_{hor} \geq l_{ver}$, ou seja, que $l_{hor} - l_{ver} \geq 0$. Assim, é garantido que se tenha largura maior ou igual à altura.

As demais funções de restrições necessárias são as mesmas tratadas pelas equações (2.19), (2.20), (2.21) e (2.22) do Capítulo 2.

3.1.3. Minimização do comprimento da aresta máxima

A *minimização do comprimento da aresta máxima* é um critério estético igualmente importante. Se o desenho adotar, por exemplo, a convenção de desenho no *grid*, este critério estético torna-se de suma importância, pois caso ele possua uma aresta muito grande, ele não poderá ser exibido inteiramente na tela do computador. A formulação por PLI que representa este critério estético é mostrada a seguir:

$$\min (\max((c_{r(e)} - c_{l(e)}), (c_{t(e),} - c_{b(e)}))) . \quad (3.6)$$

A expressão pode ser re-escrita como:

$$\min (max (l_{hor}, l_{ver})). \quad (3.7)$$

Esta expressão significa encontrar a maior aresta entre todas as arestas verticais e horizontais e então, minimizá-la.

Os modelos matemáticos que contemplam todos os critérios estéticos tratados neste trabalho, em relação à etapa de compactação, são formulados conforme mostrado nas seções anteriores. Para a solução do problema envolvendo todos os critérios estéticos modelados, será utilizada a abordagem da soma ponderada dos objetivos. Esta abordagem transforma um problema multiobjetivo em mono-objetivo e assim é possível resolvê-lo por técnicas de PLI. Se necessária a utilização de outros tipos de funções de restrições para cada critério estético, as mesmas poderão ser facilmente utilizadas na abordagem desenvolvida.

Os testes computacionais, para verificação e validação dos modelos desenvolvidos, serão detalhados nas próximas seções. Primeiramente, serão mostrados os resultados obtidos em relação à aplicação de PLI para o critério estético *minimização da soma total das arestas* e sua comparação com a abordagem do fluxo de custo mínimo em rede. Em seguida serão mostrados os resultados obtidos com a abordagem multiobjetivo aplicada aos modelos obtidos nesta seção.

3.2. Implementação computacional

A implementação computacional do problema de *compactação* considerando o critério estético "*minimização da soma total das arestas do desenho*" bem como os demais critérios, foi desenvolvida utilizando-se técnicas de programação orientada a objetos, com C++, em ambiente Windows XP/Vista com MS visual C++ 2005 / 2008. Para a abordagem que envolve técnicas de PLI, a ferramenta LPSOLVER [91] é utilizada. O LPSOLVER é uma biblioteca para a solução de problemas de programação linear, usando uma abordagem mista que envolve o algoritmo simplex e o algoritmo *branch-and-bound*. Foi também utilizada a biblioteca de desenho de grafos GTAD ([6] e [7]) que implementa, sob o paradigma de programação genérica, diversos algoritmos relacionados com desenhos de grafos (alguns deles podem ser vistos em [92], [93]). Para as etapas de *planarização* o algoritmo de planarização da GTAD recebe uma sequência que representa a ordem de inserção das arestas e devolve o embutimento planar do grafo, bem como o número de cruzamentos de arestas do grafo. Este embutimento planar é submetido ao algoritmo de ortogonalização da GTAD, que devolve a representação ortogonal do grafo e o número de dobras, e finalmente, esta representação ortogonal é submetida ao algoritmo de compactação da GTDA que devolve o grafo compactado. Porém, a GTAD não contempla técnicas de PLI e nem de otimização multiobjetivo.

3.3. Otimização multiobjetivo e desenho de grafos

Nesta seção é descrita a nova abordagem para o tratamento dos múltiplos objetivos para o passo de compactação, considerando todos os critérios estéticos relevantes para o melhoramento da qualidade (legibilidade e visibilidade) do desenho de grafos. No contexto desta nova abordagem, cada objetivo representa um critério estético a ser considerado no desenho. Diferentes critérios estéticos refletem diferentes, porém conflitantes, metas e preferências. A busca por soluções que satisfaçam aos múltiplos critérios estéticos caracteriza o problema multiobjetivo. A formulação do problema multiobjetivo está relacionada com a técnica *soma ponderada* dos objetivos, que transforma o problema com vários objetivos em um problema mono-objetivo. O problema é formulado como:

$$\min_{x \in X} \sum_{k=1}^{p} \alpha_k f_k(x) \tag{3.8}$$

Considerando quatro objetivos, temos:

$$\min F(\mathrm{x}), \text{ com } \quad F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \alpha_4 f_4(x) \quad \text{e} \quad \sum \alpha_k = 1. \tag{3.9}$$

com $\alpha_1, \alpha_2, ..., \alpha_{k1} \in R$.

Os critérios estéticos representados por cada função objetivo estão listados abaixo:

f_1 : representa *a soma total dos comprimentos das arestas*;
f_2 : representa *a área do desenho*;
f_3 : representa *a razão de aspecto*;
f_4 : representa *o tamanho da aresta mais longa.*

Minimizando os critérios estéticos citados, pode-se mostrar que é possível obter um desenho mais compacto e melhor distribuído em um dado espaço na tela de computador. Temos assim, uma nova abordagem para obtenção automática de desenho de grafos para a etapa de *compactação* da abordagem *topologia-forma-métrica*.

As expressões para cada função objetivo são mostradas a seguir:

$$f_1 = \min \sum_{e \in E_h} c_{r(e)} - c_{l(e)} + \sum_{e \in E_v} c_{t(e)} - c_{b(e)} \tag{3.10}$$

$$f_2 = \min \left((\max(c_{r(e)} - c_{l(e)})) * (\max(c_{t(e),} - c_{b(e)})) \right) \tag{3.11}$$

$$f_3 = \min (\max(\max (c_{r(e)} - c_{l(e)}), \max (c_{t(e),} - c_{b(e)})) / (\min(\max (c_{r(e)} - c_{l(e)}), \max (c_{t(e),} - c_{b(e)}))). \tag{3.12}$$

$$f_4 = \min (\max((c_{r(e)} - c_{l(e)}), (c_{t(e),} - c_{b(e)}))) . \tag{3.13}$$

Como f_2 e f_3 são funções não lineares, aplicando a abordagem para a aproximação linear mostrada na *Seção 3.1.1*, os modelos obtidos através da aproximação linear, considerando $q = 1$, são apresentados a seguir:

$$f_2 = \min\left((\max(c_{r(e)} - c_{l(e)})) + (\max(c_{t(e),} - c_{b(e)}))\right). \tag{3.14}$$

$$f_3 = \min\left(\max(\max(c_{r(e)} - c_{l(e)}), \max(c_{t(e),} - c_{b(e)})) - (\min(\max(c_{r(e)} - c_{l(e)}), \max(c_{t(e),} - c_{b(e)}))\right). \tag{3.15}$$

Na próxima seção serão apresentados os resultados relacionados a utilização de técnicas de PLI..

3.4. Resultados computacionais por PLI

Os resultados computacionais mostrados a seguir estão, primeiramente, relacionados ao tratamento do critério estético "*minimização da soma total das arestas do desenho*", utilizando técnica de PLI. Estes resultados são comparados aos da abordagem clássica para o mesmo grafo de teste.

Para todos os experimentos executou-se as três etapas da abordagem *topologia-forma-métrica* da seguinte maneira: na *planarização*, foi utilizada a abordagem tradicional (abordagem incremental para a construção do embutimento planar fixo detalhada no Capítulo 2); na *ortogonalização*, foi utilizado o modelo por PLI para obtenção da representação ortogonal ***H*** (visto no capítulo 2) e na *compactação*, foi utilizado tanto a abordagem do fluxo de custo mínimo em rede como a abordagem por PLI para o critério estético "*minimização da soma total das arestas do desenho*". A seguir são mostradas duas figuras que ilustram os resultados obtidos, uma utilizando a abordagem do fluxo de custo mínimo em redes na compactação e a outra utilizando PLI. Assim, as *Figuras 3.2* e 3.*3* mostram dois desenhos compactados para o mesmo grafo *G*. O primeiro (Figura 3.2) foi compactado pela técnica de compactação tradicional (fluxo de custo mínimo em rede) e o segundo (Figura 3.3), utilizando a técnica de PLI descrita. Na *Figura 3*.2, que mostra um desenho de um grafo compactado utilizando técnicas baseadas na abordagem do fluxo de custo mínimo em rede, pode-se observar que foi gerado um desenho cujo tamanho total das arestas não representa um desenho ótimo para o caso mostrado. Na *Figura 3*.3, o mesmo desenho é mostrado, porém, agora compactado utilizando técnicas de PLI. Neste caso, o desenho é mais otimizado. Com isto, pode-se observar que a *compactação,* utilizando técnicas de PLI, mostra resultados otimos, considerando o critério estético "*minimização da soma total dos tamanhos das arestas"*.

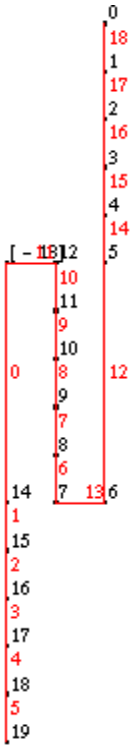


Figura 3. 2, Exemplo de um grafo compactado utilizando a abordagem do fluxo.

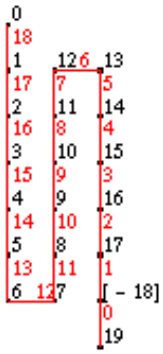


Figura 3. 3 -- Exemplo do mesmo grafo utilizado na Figura. 3.2, desta vez compactado utilizando técnicas de PLI.

Nesta etapa do trabalho, foi possível testar e observar a utilização da abordagem para compactação, identificada como abordagem dos segmentos (detalhada no Capítulo 2), para um desenho ortogonal no *grid*. Fica claramente observada a melhoria obtida com a abordagem por PLI, em relação a abordagem clássica (*topologia-forma-métrica*). Isto abre caminho para o desenvolvimento da nova abordagem com base em técnicas de PLI.

Uma vez testada a abordagem dos segmentos, utilizando-se técnicas de PLI para o caso em que se considera apenas um critério estético e de posse dos modelos que representam os demais critérios estéticos (funções objetivo), técnicas de otimização multiobjetivo são

utilizadas considerando todos os critérios estéticos, simultaneamente. De acordo com a formulação dada pela *Equação* (*3.9*), o problema é resolvido pela soma ponderada dos múltiplos objetivos. Esta técnica permitiu validar os modelos desenvolvidos e avaliar os resultados obtidos em termos de desenho. Vale lembrar que não existe na literatura nenhum indicativo de solução do problema de *compactação* da abordagem *topologia-forma-métrica*, considerando mais de um objetivo. Portanto, esta abordagem é inédita neste contexto.

Os resultados obtidos com esta nova abordagem são mostrados, considerando inicialmente o exemplo de grafo ilustrado a seguir.

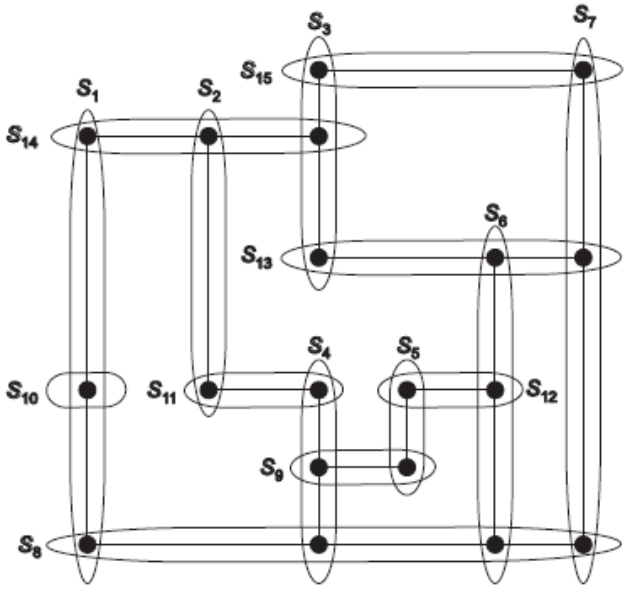


Figura 3.4 –Desenho inicial (*19 vértices e 23 arestas*) *usando a abordagem do segmento para uma representação ortogonal.*

A *Figura 3.4* mostra um exemplo de um grafo ortogonal, o qual será utilizado para ilustrar a abordagem. Nesta figura, s_1 é o comprimento do segmento s_1 e é obtido pela diferença entre o valor da coordenada *y* do segmento s_{14} e o da coordenada *y* do segmento s_8. Para esta ilustração foi utilizado um grafo com 19 vértices e 23 arestas.

Usando a abordagem do segmento (discutida no capítulo 2), foram obtidas as seguintes funções objetivo que caracterizam o exemplo da *Figura 3.4* e que representam os critérios estéticos tratados neste trabalho, para a etapa de compactação:

f_1 = *soma total dos tamanhos das arestas:*

$$f_1 = 2s_{14} - 4s_8 + 2s_{15} + s_{12} - s_9 + 3s_7 - 2s_1 - s_2 + s_6 - s_3 . \quad (3.16)$$

f_1 é obtida pela soma de todos os segmentos do exemplo da *Figura 3.4*:

$$f_1 = s_1 + s_2 + \ \ldots \ + s_{15} \ . \tag{3.17}$$

Na *Equação* (*3.17*), cada segmento s_i da *Figura 3.4* é substituído pela diferença entre as coordenadas *x* do segmento à estrema direita de s_i e do segmento à extrema esquerda de s_i se s_i for um segmento horizontal, ou entre as coordenadas *y* do segmento acima de s_i e o segmento abaixo de s_i se s_i for um segmento vertical, por exemplo: $s_1 = s_{14} - s_8$; $s_2 = s_{14} - s_{11}$; $s_8 = s_7 - s_1$ e assim sucessivamente. Após a substituição de todos os s_i's na *Equação* (*3.17*), a mesma é simplificada e a função objetivo f_1 *Equação* (*3.16*) é obtida. Da mesma maneira é feito para obtenção de f_2, f_3 *e* f_4, de acordo com seus modelos.

f_2 = *a área do desenho:* Neste caso, a aproximação linear mostrada na *Seção 3.1.1* é utilizada.

$$f_2 = s_{15} - s_8 + s_7 - s_1 \ . \tag{3.18}$$

f_3 = *razão de aspecto do desenho:* Neste caso, é também utilizada a aproximação linear.

$$f_3 = s_7 - s_1 - s_{15} + s_8 \ . \tag{3.19}$$

f_4 = *tamanho da maior aresta (aresta mais longa):*

$$f_4 = s_7 - s_1 \ . \tag{3.20}$$

Usando a abordagem *soma ponderada*, a função objetivo $F(x)$ que combina todos os critérios estéticos, é dada pela *Equação* (*3.9*).

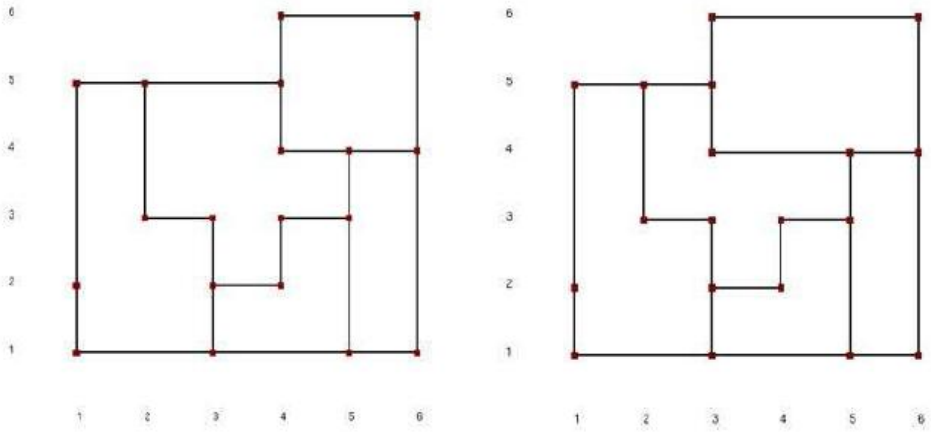
A minimização de $F(x)$ foi resolvida utilizando técnicas de PLI. A seguinte função de restrição foi adicionada para assegurar que o desenho resultante terá a largura maior ou igual à altura, conforme restrição da *Equação* (*3.4*):

$$s_7 - s_1 - s_{15} + s_8 \geq 0 \tag{3.21}$$

Os resultados obtidos com esta nova abordagem, utilizando a soma ponderada dos objetivos, são mostrados na *Tabela 3.1* e serão ilustrados com figuras para cada caso mostrado na tabela. RA é o valor da razão de aspecto para cada caso e os valores dos pesos (α) podem ser vistos também na tabela.

Tabela 3.1 – Tabela de resultados das funções objetivo para o exemplo da *Figura 3.4* (grafo com 19 vértices e 23 arestas).

caso	α_1	α_2	α_3	α_4	f_1	f_2	f_3	f_4	$F(x)$	RA
3a	1	0	0	0	34	10	0	5	34	1.0
3b	0	1	0	0	35	10	0	5	10	1.0
3c	0	0	1	0	38	11	1	6	1	1.2
3d	0	0	0	1	35	10	0	5	5	1.0
3e	0.3	0.7	0	0	34	10	0	5	17.2	1.0
3f	0	0.3	0.7	0	39	11	1	6	4	1.2
3g	0	0	0.3	0.7	39	11	1	6	4.5	1.2
3h	0.3	0	0	0.7	34	10	0	5	13.7	1.0
3i	0.3	0.2	0.3	0.2	36	11	1	6	14,5	1.2
3j	0.3	0.2	0.3	0.2	34	10	0	5	13.2	1.0



(a) (b)

Figura 3.5 – Desenho ortogonal usando otimização mono-objetivo para cada critério estético: (a) a soma total dos tamanhos das arestas (caso 3a), (b) área do desenho (caso 3b).

A *Figura 3.5* está associada ao caso 3a e 3b da *Tabela 3.1*. Pode-se observar que o mínimo para a *soma total dos tamanhos das arestas* nas *Figura 3.5* (a) é 34 e a área é 10. Neste caso, foi ponderado somente o objetivo f_1 (*soma total dos tamanhos das arestas*). Na *Figura 3.5* (b) a soma total das arestas é 35, o que é pior que o obtido no caso 3a. A área é a mesma (10), porque, para este exemplo, este é o seu mínimo. Neste caso foi ponderado somente f_2 (área).

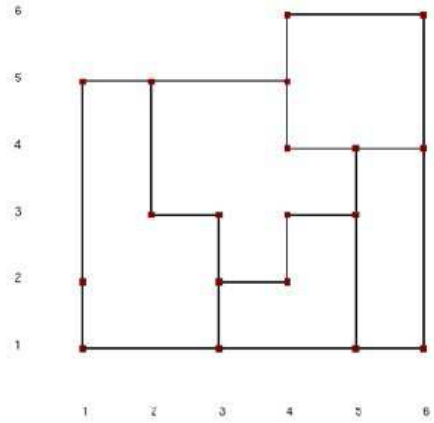
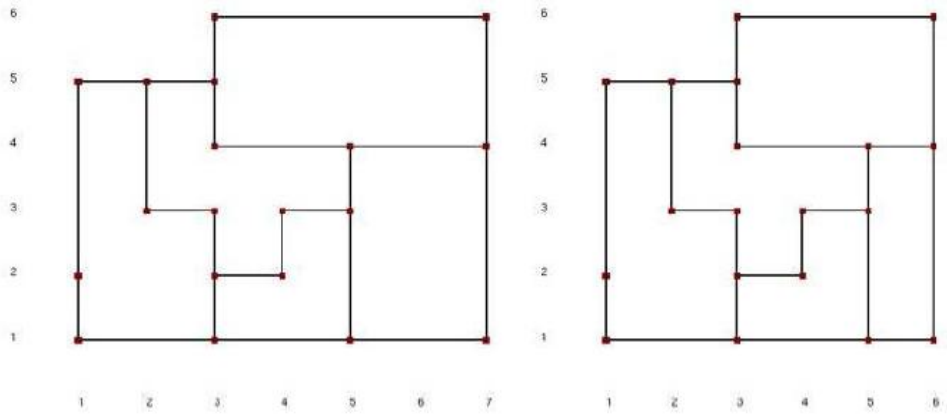


Figura 3.6 – Desenho ortogonal combinando os critérios estéticos: a soma total dos comprimentos das arestas e a área do desenho, caso 3e, com $\alpha_1 = 0.3$ e $\alpha_2 = 0.7$.

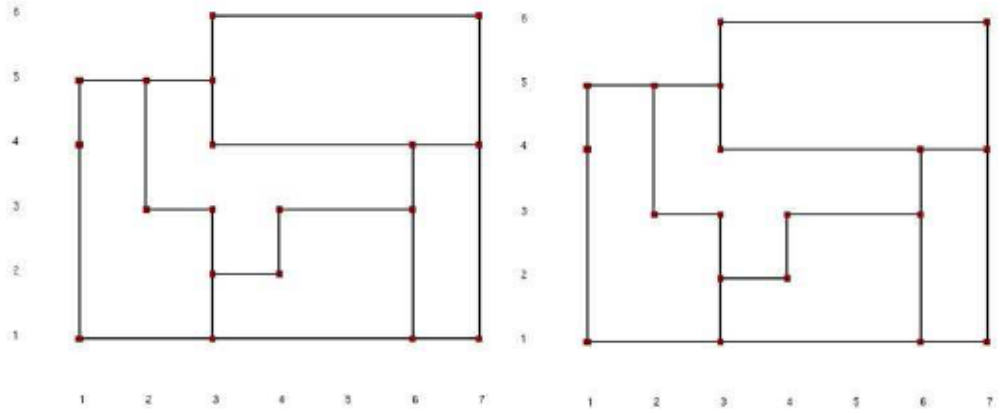
A *Figura 3.6* está associada com o caso *3e* na *Tabela 3.1*, em que foram combinados dois objetivos: *minimização da soma total dos tamanhos das arestas* e a *minimização da área do desenho*. Neste caso, foram considerados os dois objetivos simultaneamente.



(a) (b)

Figura 3.7 – Desenho ortogonal utilizando otimização multiobjetivo para os critérios estéticos: (a) melhor razão de aspecto (caso 3c) e (b) melhor tamanho da aresta mais longa do desenho (caso 3d).

Para a situação mostrada na *Figura 3.7* que representa o caso 3c, no qual se considera somente o critério estético *razão de aspecto*, é considerada a função de restrição da *Equação* (*3.21*). Isto garante que o desenho será melhor distribuído e melhor visualizado em diversos monitores de computador. Pode-se notar que neste caso não é possível minimizar a *soma total dos tamanhos das arestas*, cujo valor é 38, ou a *área*, o que é natural, uma vez que está restrito a largura em relação à altura do desenho. No caso *3d* temos o melhor desenho para o objetivo *minimizar o tamanho da aresta mais longa*. Assim, é possível mostrar que pode-se aplicar as restrições necessárias ao modelo utilizando-se de técnicas de PLI.



(a) (b)

Figura 3.8 – Desenho ortogonal utilizando otimização multiobjetivo para os critérios estéticos: (a) melhor desenho combinando: a área e a razão de aspecto com $\alpha_2 = 0.3$ e $\alpha_3 = 0.7$ *(caso 3f). (b) melhor desenho combinando a razão de aspecto e o tamanho da aresta mais longa do desenho com* $\alpha_3 = 0.3$ e $\alpha_3 = 0.7$ *(caso 3g).*

O grafo da *Figura 3.8* (a) está associado com o caso *3f* da *Tabela 3.1*, na qual foram combinados dois objetivos (f_2 e f_3). O grafo em (b) está associado com o caso *3g*, em que também foram combinados dois objetivos (f_3 e f_4). Nestes casos, a *área* e o *tamanho da maior aresta* não atingiram o seu mínimo e seus valores são iguais para os dois casos. Por isto, suas figuras são iguais. É importante lembrar que a função de restrição da *Equação* (*3.21*) utilizada neste caso, limita melhores resultados para a *área*, e consequentemente, para o *tamanho da maior aresta* também, o que é natural nesta situação.

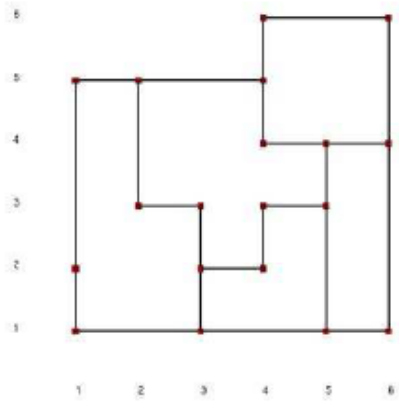
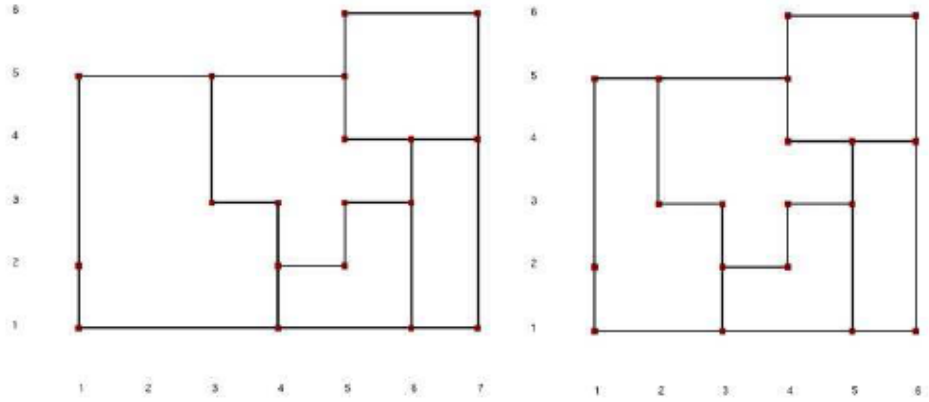


Figura 3.9 – Desenho ortogonal combinando os critérios estéticos: a soma total dos tamanhos das arestas e o tamanho da aresta mais longa do desenho) com $\alpha_1 = 0.3$ e $\alpha_4 = 0.7$ *(caso 3h).*

.

Para o grafo da *Figura 3.9* (caso *3h*) e os valores de α utilizados, tem-se o melhor desenho quando são combinadas f_1 e f_4, simultaneamente.



(a) (b)

Figura 3.10 – Desenho ortogonal utilizando otimização multiobjetivo para os critérios estéticos: (a) o melhor desenho combinando todas as funções objetivo para os valores de α usados (caso 3i). (b) o melhor desenho combinando todas as funções objetivo sem a função de restrição da equação (3.21) para os valores de α usados (caso 3j).

A *Figura 3.10* (a) mostra o melhor resultado obtido combinando todos os quatro objetivos (f_1, f_2, f_3 e f_4), para os valores de α utilizados. Esta é a ilustração do caso 3*i* na *Tabela 3.1*. Nesta situação, pode-se observar que não se tem o melhor resultado comparando com alguns casos individuais, mas se tem um desenho otimizado considerando todas as funções objetivo, simultaneamente (neste caso foi considerada a função de restrição da *Equação* (*3.21*)).

Na *Figura 3.10* (b) é mostrada a mesma situação da *Figura 3.10* (a), mas sem considerar a função de restrição da *Equação* (*3.21*), o que torna o desenho o mais compacto possível. Pode-se observar que se tem um desenho otimizado, considerando todos os critérios estéticos modelados. Com isto, todos estes resultados mostram que se pode obter desenhos otimizados buscando uma boa relação de compromisso entre diferentes critérios estéticos, o que oferece flexibilidade para a escolha das funções objetivo e funções de restrições que melhor representam a aplicação, ou mesmo as necessidades dos usuários, ou o especialista da aplicação.

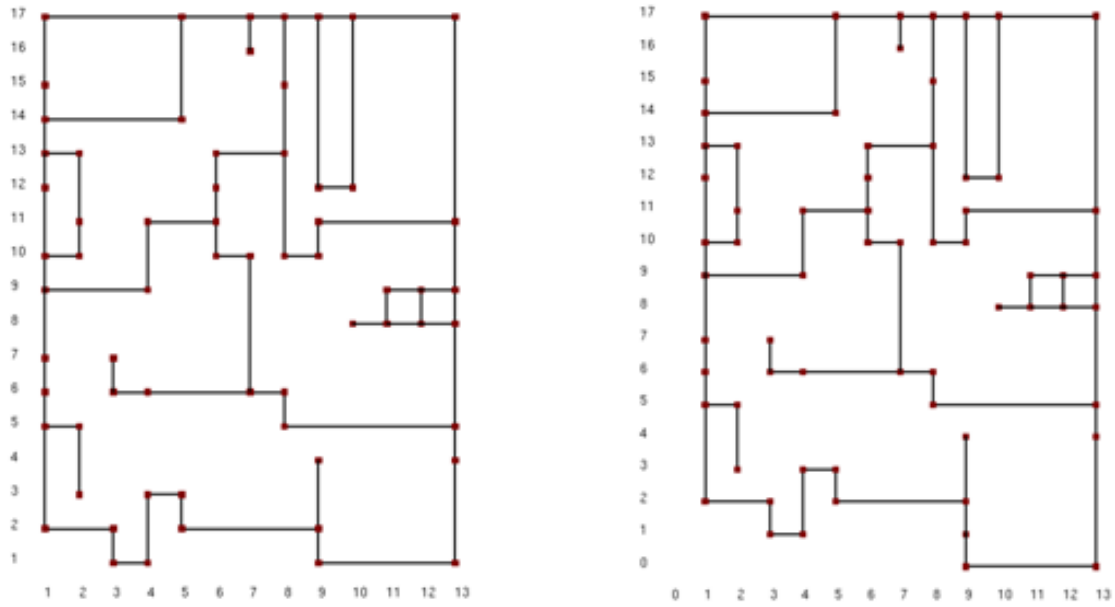
Para o exemplo utilizado, o desenho para os casos *3b* e *3d* são iguais, Isto sugere uma equivalência entre as funções objetivo f_1 e f_4. Porém, para exemplos maiores, os desenhos mostram diferenças, mesmo que sutis. Esta situação está ilustrada na *Figura 3.11*, onde o exemplo utilizado foi um grafo com 69 vértices.

Não foi possível testar para exemplos maiores, pois a implementação de uma maneira de se obter as funções objetivo de forma automática, por não ser trivial, mostrou dificuldades grandes e não foi atingida nesta trabalho.

A *Tabela 3.2* mostra resultados obtidos para um exemplo com 69 vértices e 79 arestas (*Figura 3.11*) em relação ao exemplo da *Figura 3.4*.

Tabela 3.2 – Tabela de resultados das funções objetivo para um exemplo de um grafo com 69 vértices e 79 arestas.

Caso	α_1	α_2	α_3	α_4	f_1	f_2	f_3	f_4	**F**	**RA**
4a	1	0	0	0	162	28	4	16	134	1.33
4b	0	1	0	0	172	28	4	16	28	1.33
4c	0	0	1	0	214	34	0	17	0	1.0
4d	0	0	0	1	174	29	5	17	12	1.42
4e	0.3	0.7	0	0	162	28	4	16	59.8	1.33
4f	0	0.3	0.7	0	195	32	0	16	9.6	1.0
4g	0	0	0.3	0.7	195	32	0	16	11.2	1.0
4h	0.3	0	0	0.7	162	28	4	16	48.6	1.33
4i	0.3	0.2	0.3	0.2	178	32	0	16	54.6	1.0
4j	0.3	0.2	0.3	0.2	162	28	4	16	47	1.33



(a) (b)

Figura 3.11 – Desenho ortogonal utilizando otimização multiobjetivo para os critérios estéticos: (a) melhor área do desenho (caso 4b) e (b) tamanho da aresta mais longa do desenho (caso 4d).

A *Figura 3.11* ilustra a diferença em termos de desenho para os critérios estéticos relacionados com a *minimização da área* do desenho e a *minimização da aresta mais longa* do mesmo (extremidade inferior direita do desenho (b)).

Até este momento, esforços foram dedicados à aplicação de técnicas de PLI na etapa de *compactação* do desenho de grafos da abordagem *topologia-forma-métrica*. Para isto foi necessário avaliar, implementar e testar a abordagem sugerida na literatura que modela o

problema de desenho de grafos como um problema de PLI. Foi possível modelar, com base na abordagem dos segmentos, os demais critérios estéticos relevantes para a etapa de compactação da abordagem *topologia-forma-métrica* como um problema de PLI. Como são vários os critérios estéticos, sendo que alguns deles conflitam entre si, foi possível resolver o problema multiobjetivo utilizando técnicas desta natureza. Os modelos com base nestas técnicas foram implementados e testados com sucesso. Seus resultados foram mostrados e discutidos nas seções anteriores. Esta nova abordagem permite maior flexibilidade para a utilização de diversos critérios estéticos e restrições de desenho, de acordo com as necessidades dos usuários ou especialistas da aplicação. Porém existem ainda algumas dificuldades: a definição dos pesos ideais para a soma ponderada dos objetivos; a obtenção das funções objetivos de forma automática; a aproximação linear utilizada nem sempre garante resultados efetivamente próximos do real. Assim, o objetivo proposto para esta etapa do trabalho foi atingido com as ressalvas citadas . Os resultados foram mostrados. Não foi possível realizar testes para grafos maiores, em função da difuldade encontrada no desenvolvimento computacional de uma técnica que permitisse a obtenção, de forma automática, das funções objetivo, dado um grafo qualquer. Esta tarefa será indicada para desenvolvimento em trabalhos futuros. Além disto, é interessante avaliar também se é possível melhorar a aproximação linear utilizada.

Na próxima seção serão tratados os resultados obtidos através de técnicas de otimização evolucionária.

3.5. Técnicas evolucionárias em desenho de grafos

Como já discutido anteriormente, *minimizar cruzamentos*, *minimizar número de dobras*, entre outros critérios estéticos, são problemas da classe NP-Difícil. Assim, heurísticas são utilizadas para se obter resultados aproximados. Porém, o sucesso na utilização das heurísticas é influenciado também pelas dificuldades comuns em desenho de grafos, entre elas, o tratamento de instâncias de grafos muito grandes requer um tempo de processamento muito elevado. Em problemas de otimização combinatória o espaço de busca é muito extenso e exige um tratamento minusioso de forma a explorar toda a sua extensão. Por isto, neste trabalho, metodologias são desenvolvidas com base em meta-heurísticas, utilizando algoritmos evolucionários (algoritmo genético) que permite explorar este espaço de busca de forma eficiente [42], [43], [44], [45], [46] e [47], visando resolver os seguintes problemas em aberto:

- Poder avaliar um número *m* de embutimentos planares na definição da topologia do desenho, de maneira a avaliar se é possível obter desenhos mais otimizados ao final do processo;

- Poder tratar o problema de obtenção de desenhos ortogonais no *grid*, em apenas uma etapa.

Visando a atender aos objetivos citados, são modeladas e implementadas as técnicas de otimização com base na abordagem evolucionária (algoritmo genético). O primeiro dos objetivos acima é tratado neste capítulo e o segundo no próximo capítulo. Os resultados obtidos são analisados em termos de números, bem como os desenhos resultantes. O enfoque considerado nesta análise está relacionado com a qualidade do desenho, o que se traduz em visibilidade e legibilidade do mesmo. Sua qualidade deve ser avaliada, lembrando que a qualidade de um desenho em termos de critérios estéticos, está relacionada com o quão satisfatório é o critério estético ou o conjunto de critérios estéticos tratados. Nesta etapa, visa-se resolver o problema em aberto relacionado com a utilização de variados embutimentos planares a serem submetidos às demais etapas da abordagem *topologia-forma-métrica*. O objetivo é obter um desenho final melhor, a partir de um conjunto de embutimentos planares. Como o algoritmo genético é aplicado na etapa de *planarização* e seu resultado é submetido às etapas de *ortogonalização* e *compactação* pela abordagem clássica (*topologia-forma-métrica*), neste senso, pode-se considerar esta abordagem como uma abordagem híbrida utilizando a *topologia-forma-métrica* e o *algoritmo genético*.

Além da abordagem híbrida, é também desenvolvida a abordagem unificada, que utilizando o algoritmo genético, visa tratar do problema de desenho de grafos em apenas uma etapa. Esta será tratada no próximo capítulo. Neste caso, é aplicado o algoritmo genético de forma a tratar o problema de desenho automático de grafos em uma única etapa, visando eliminar a interdependência entre as etapas existente na *topologia-forma-métrica*. Todos os critérios estéticos são considerados, simultaneamente. Logo, esta etapa do trabalho foi dividida em duas abordagens que serão identificadas como abordagem híbrida e abordagem unificada.

3.6. Abordagem híbrida

Uma abordagem híbrida é desenvolvida para desenho automático de grafos com base na *topologia-forma-métrica* e o uso do *algoritmo genético*. O problema de buscar por melhores embutimentos planares na etapa de planarização é formulado como um problema de otimização combinatória baseado em permutação de inteiros [37], [38] e [39]. É empregado o algoritmo genético para resolver este problema baseado em permutação. O AG trabalha com uma população de *N* indivíduos que são representados por permutação de inteiros. Cada permutação representa a sequência de inserção das arestas pelo algoritmo que gera o embutimento planar na planarização. Desta maneira, cada permutação

representa um embutimento planar diferente, permitindo assim a obtenção de *m* embutimentos planares. Consequentemente, o *i*-ésimo indivíduo da população na geração *t,* representado por $S_{t,i}$, é uma permutação de inteiros com valores a partir de 1 até o número total de arestas do grafo. Com esta abordagem é possível explorar um maior número de embutimentos planares na etapa de planarização e submeter todos eles às etapas de *planarização, ortogonalização* e *compactação,* de modo a avaliar a qualidade de cada embutimento planar. Para se atingir os objetivos definidos, foram desenvolvidas três metodologias considerando três abordagens, a saber:

I. Na primeira, o problema multiobjetivo considerando os três objetivos conflitantes já citados foi transformado em um problema mono-objetivo e foi resolvido pela soma ponderada (TSM-WS) dos três objetivos, conforme a *equação 3.22* (*Seção 3.7*). Esta soma ponderada foi utilizada como função de aptidão no processo evolucionário do algoritmo genético;
II. Na segunda, o modelo *fuzzy para tomada de decisão multicritério* foi utilizado no processo evolucionário do algoritmo genético (TSM-*FUZZY*), bem como na escolha da solução após as 10 execuções do AG, garantindo, assim, soluções mais harmoniosas;
III. Na terceira e última foi utilizado o algoritmo genético multiobjetivo, NSGAII, para a obtenção do conjunto solução de Pareto (TSM-NSGAII). Neste caso, foi utilizado também o modelo *fuzzy de tomada de decisão multicritério* para escolha de uma solução dentre as diversas soluções no conjunto Pareto-ótimo resultante de cada execução do AG. Foi aplicada esta metodologia também para escolha de uma solução, considerando as 10 execuções do AG, para cada um dos casos de teste. Desta forma são garantidas soluções finais mais harmoniosas.

O diagrama de bloco representando os principais passos da *abordagem topologia-forma-métrica* é mostrado na *Figura 3.25*.

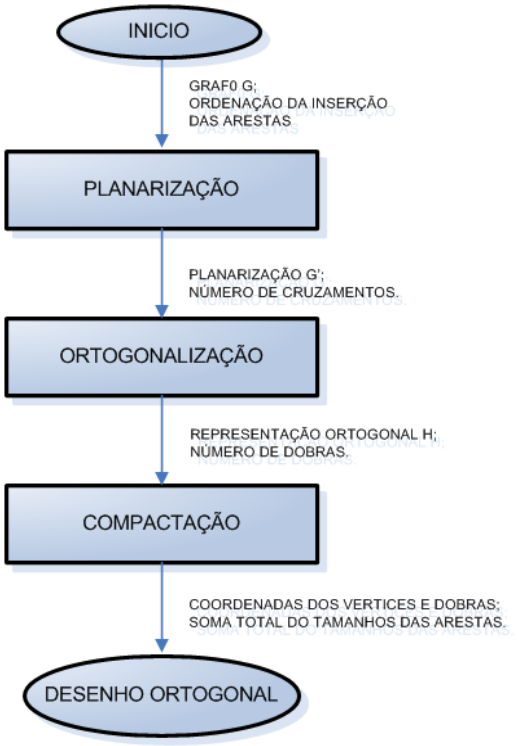


Figura 3.25 – Diagrama de blocos representando a abordagem topologia-forma-métrica.

São aplicados os operadores de recombinação/cruzamento e mutação para representação por permutação descritos a seguir:

A) Representação por permutações: Muitos eventos são, naturalmente, do tipo onde é necessário decidir a ordem na qual a sequência dos eventos deve ocorrer. Para este tipo de problema, a representação mais natural é a chamada representação por permutação de um conjunto de números inteiros. Enquanto em um algoritmo genético, com base em ordenação, os números que compõem uma *string* podem ocorrer mais que uma vez. No problema em estudo, este tipo de sequência de inteiros não será considerada como uma representação válida. Cada embutimento do grafo planar é obtido de acordo com a ordem em que as arestas são inseridas para a formação do embutimento. Dois embutimentos são considerados diferentes se a ordenação de inserção das arestas forem diferentes. Assim, será necessária a utilização de operadores que preservem as propriedades da permutação, ou seja, garantir que determinadas sequências de valores ocorrerão exatamente uma vez na solução. Estes operadores serão discutidos a seguir:

B) Operação de mutação para representação por permutações: Para representações por permutações não é possível considerar cada gene independentemente, então encontrar mutações apropriadas é um problema de movimentação de alelos dentro do genoma. A consequência imediata disto é que o parâmetro de mutação é interpretado como uma probabilidade que a *string* sofra mutações de maneira que um único ou mais

genes na *string* sejam alterados. As três maneiras de mutação mais comuns utilizadas em problemas baseados em ordenação serão descritas a seguir.

Neste contexto: o genoma é representado pela *string* inteira; o gene, por cada posição da *string* e o alelo, pelo valor em cada posição.

Operador de mutação por troca (SWAP): Este operador troca aleatoriamente duas posições (genes) na *string*, trocando os valores de seus alelos. Veja a *Figura 3.12* para ilustração:

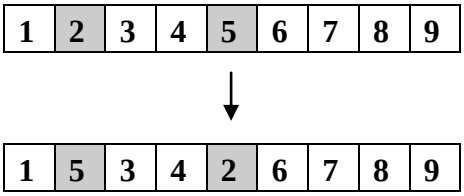


Figura 3.12 – Mutação por troca

Operador de mutação por inserção: Este operador funciona de maneira a escolher dois alelos aleatoriamente e mover um para próximo do outro, abrindo espaço entre os demais alelos. Veja a ilustração na *Figura 3.13*.

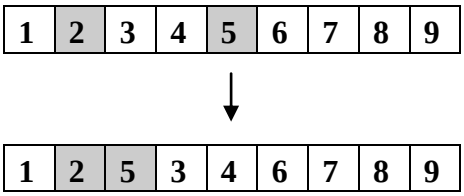


Figura 3.13 – Mutação por inserção

Operador de mutação por scramble: Este operador funciona de maneira a escolher aleatoriamente um subconjunto ou toda a *string* e reorganizar a posição dos alelos na *string* ou suconjunto dela também aleatoriamente. Veja a ilustração na *Figura 3.14*.

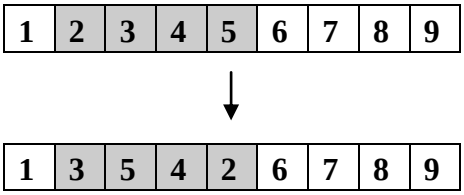


Figura 3.14 – Mutação por scramble.

Operador de mutação por inversão: Este operador funciona de maneira a escolher aleatoriamente um subconjunto ou toda a *string* e inverter a posição dos alelos na *string* ou subconjunto dela. Veja a ilustração na *Figura 3.15*.

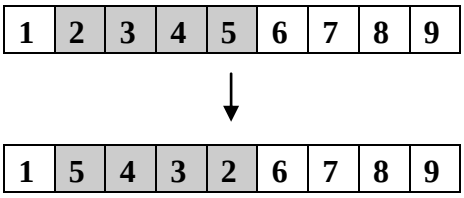


Figura 3.15 – Mutação por inversão.

C) Operação de recombinação ou cruzamento para representação por permutações: O processo de recombinação ou cruzamento é aquele em que as informações de dois ou mais indivíduos são utilizadas para se criar um novo individuo/descendente. Assim, este novo indivíduo leva consigo características de seus pais. Esta é a operação considerada como a mais importante para os algoritmos genéticos [37].

Representações baseadas em permutaçoes apresentam uma dificuldade particular para o caso dos operadores de recombinação/cruzamentos, uma vez que não é geralmente possível fazer trocas simples de substrings dos pais e ainda manter as propriedades de permutação. Mas o problema é resolvido através de um número de operadores especializados, os quais foram desenvolvidos para se trabalhar com representações por permutações. Estes operadores visam transmitir, o tanto quanto possível, as informações dos pais para seus descendentes. Serão descritos a seguir dois dos mais conhecidos e mais utilizados operadores para cada subclasse de problemas:

Operador de recombinação/cruzamento parcialmente mapeado (PMX): Cruzamento parcialmente mapeado é o operador mais utilizado para problemas de *adjacências*. Desde que foi proposto, ele sofreu diversas variações. Será apresentada aqui uma de suas variações mostrada em [37] e que segue os seguintes passos:

1. Escolha dois pontos de cruzamento aleatoriamente e copie o segmento entre eles a partir do primeiro pai (P1) para o primeiro descendente;
2. Partindo do primeiro ponto de cruzamento, procure por elementos naquele segmento, porém no segundo pai (P2), que não tenham sido copiados;
3. Para cada um destes (*i*) procure no descendente e veja que elemento (*j*) foi copiado em seu lugar a partir de P1;
4. Coloque *i* na posição ocupada por *j* em P2, desde que se saiba que não será colocado o *j* neste lugar (caso já o tenha na string);
5. Se o lugar ocupado por *j* em P2 já tenha sido preenchido no descendente por um elemento *k*, ponha *i* na posição ocupada por *k* em P2;

6. Uma vez feito para todos os elementos do segmento de cruzamento, o resto do descedente pode ser preenchido a partir de P2; e o segundo descendente é criado analogamente.

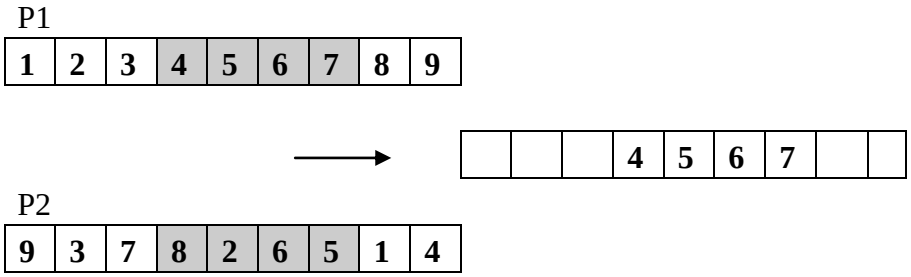


Figura 3.16 – Cruzamento parcialmente mapeado – passo 1.

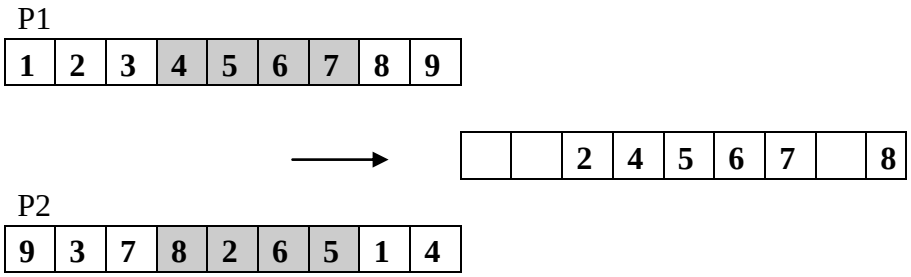


Figura 3.17 – Cruzamento parcialmente mapeado – passo 2.

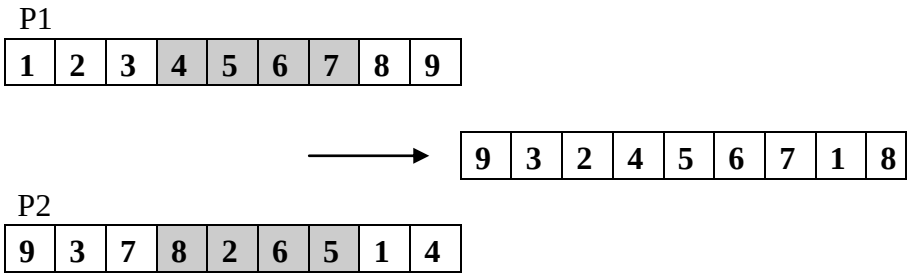


Figura 3.18 – Cruzamento parcialmente mapeado – passo 3.

Operador de recombinação/cruzamento por ordenação (OX): o operador de cruzamento por ordenação foi desenhado para problemas de permutação baseados em ordem. O seu procedimento inicial é bastante parecido com o operador parcialmente mapeado, no que tange à cópia do segmento aleatoriamente escolhido para o seu descendente. A partir daí, o procedimento se diferencia porque a intenção é transmitir informações sobre a ordem relativa do segundo pai:

1. Escolha aleatoriamente dois pontos de cruzamento e copia-se o segmento entre eles para o descendente, a partir do primeiro pai (P1);
2. Partindo do segundo ponto de cruzamento no segundo pai (P2), copie os números não usados, remanescentes, para o primeiro filho na ordem em que eles aparecem no segundo pai, colocando-os no final da lista;

3. Crie o segundo descendente de maneira análoga, com os papéis de pais invertidos.

Exemplos para estes procedimentos são ilustrados nas *Figura 3.19* e *3.20*.

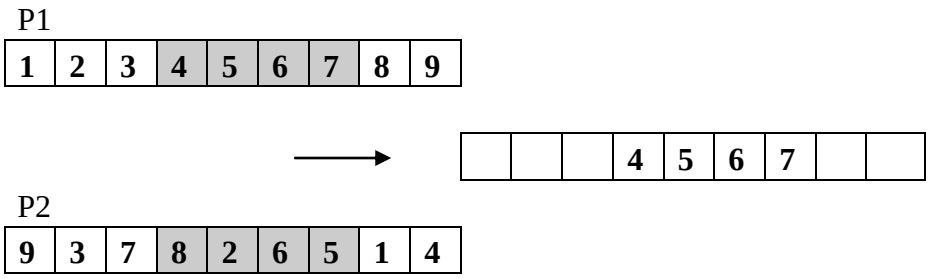


Figura 3.19 – Cruzamento por ordem – passo 1.

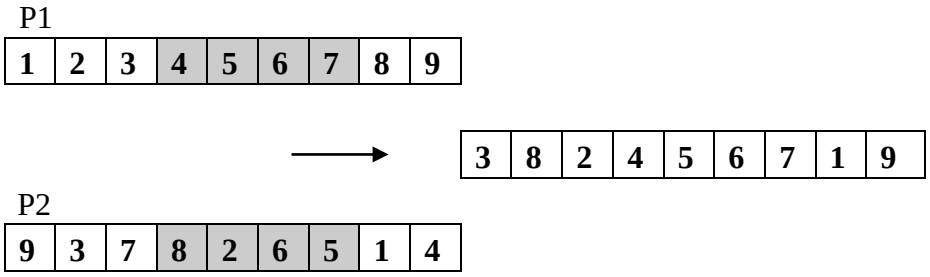


Figura 3.20 – Cruzamento por ordem – passo 2.

Os operadores citados são avaliados, implementados e aplicados neste trabalho.

D) Passos para a abordagem híbrida

Desta maneira os passos para a *abordagem híbrida*) são resumidos a seguir:

- dada a sequência de representação, provoca-se variações na ordenação de inserção das arestas e com isto obtem-se variados embutimentos planares;
- para um conjunto de embutimentos planares obtidos, aplica-se a *planarização*, *ortogonalização* e a *compactação*;
- avalia-se o grau de satisfação dos critérios estéticos desejados e escolhe o desenho que melhor represente todos os critérios estéticos simultaneamente. É utilizado o algoritmo genético para escolha destes embutimentos planares otimizados.

Será mostrado um exemplo para ilustrar a representação por permutação na obtenção de novos embutimentos planares do grafo G1, bem como um operador de mutação e um operador de cruzamento estudado. Veja *Figura 3.21*.

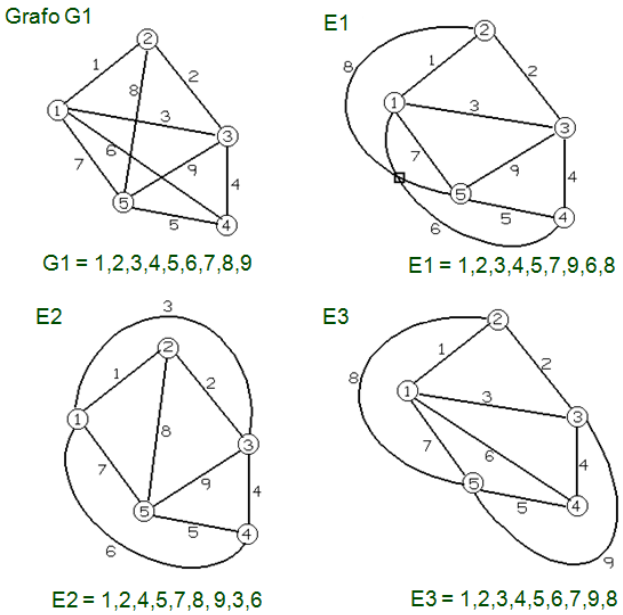


Figura 3.21 – Exemplo de um grafo G1 e três embutimentos planares obtidos variando a ordenação da inserção de suas arestas.

As representações por permutações de G1, bem como dos embutimentos planares E1, E2 e E3, são mostrados a seguir:

$\sum_{i=1}^{A} C_r$ representa o somatório dos cruzamentos em um embutimento planar, ou seja, é a função objetivo da *minimização de cruzamentos* para um embutimento planar, sendo A o conjunto das arestas do embutimento do grafo planar.

G1

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

$\sum_{i=1}^{A} C_r = 3;$

E1

1	**2**	**3**	**4**	**5**	**7**	**9**	**6**	**8**

$\sum_{i=1}^{A} C_r = 1;$

E2

1	**2**	**4**	**5**	**7**	**8**	**9**	**3**	**6**

$\sum_{i=1}^{A} C_r = 0;$

E3

1	2	3	4	5	6	7	9	8

$\sum_{i=1}^{A} C_r = 0;$

A aplicação do operador de *mutação por scramble* e o *operador de cruzamento parcialmente mapeado* para o exemplo da *Figura 3.21* é mostrado a seguir:

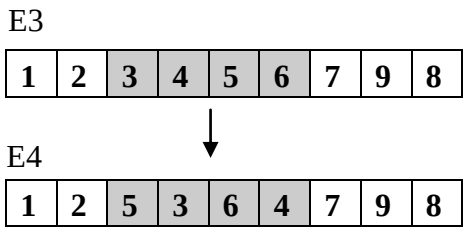


Figura 3.22 – Exemplo de aplicação do operador de mutação por scramble em um embutimento do grafo G1, gerando um novo embutimento.

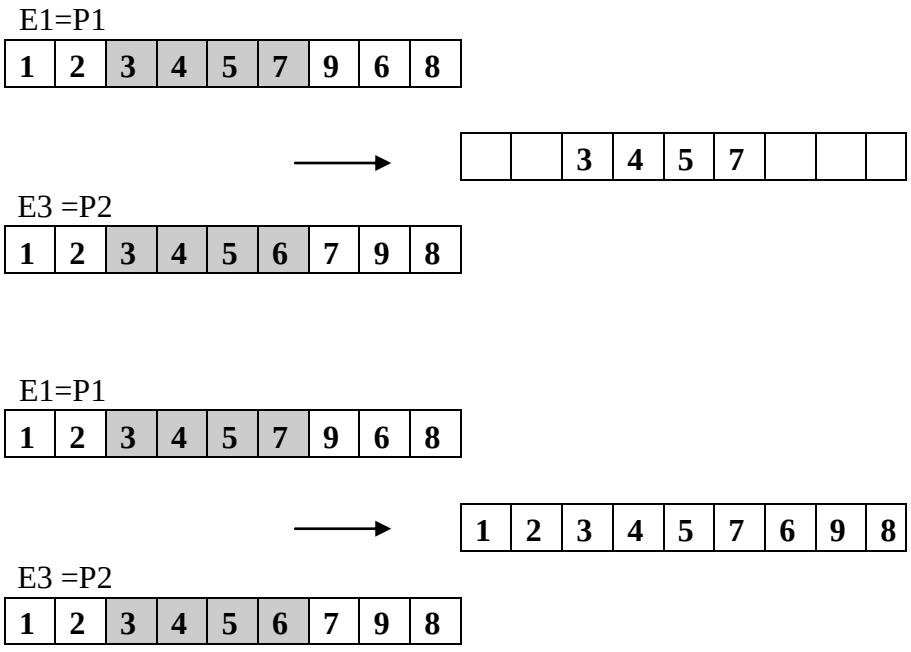
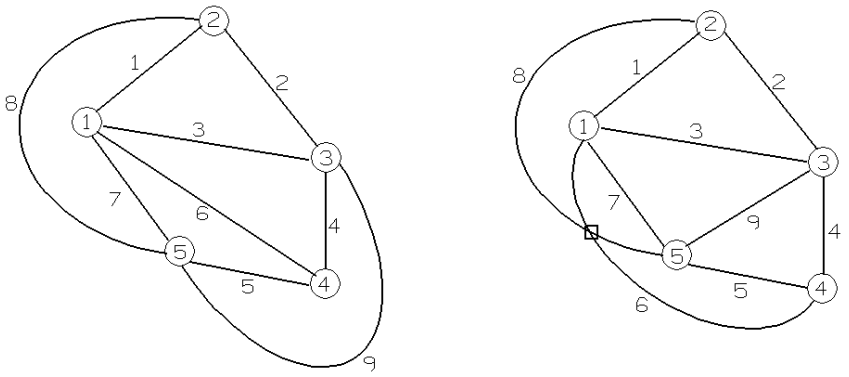


Figura 3.23 – Exemplo da aplicação do operador de cruzamento parcialmente mapeado a partir de dois embutimentos planares do grafo G1 (E1, E3), obtendo assim um novo descendente.



(a) *(b)*

Figura 3.24 – Embutimento planar obtido: (a) após aplicação do operador de mutação (Figura 3.22) e (b) após aplicação do operador de cruzamento (Figura 3.23).

No exemplo foi ilustrado que é possível obter embutimentos planares diferentes em função da aplicação dos operadores de recombinação e mutação, utilizando a representação com base em permutações de inteiros e o algoritmo genético, conforme mostrado.

É também importante ressaltar que trabalhos que utilizam algoritmos genéticos para desenho de grafos são encontrados na literatura, porém, na sua maioria, é considerada a abordagem hierárquica para desenho de grafos [94], [95], [96], [97]. Outros consideram o desenho de grafos na convenção de desenho em linhas retas [98], [99]. Investigações sobre o uso de algoritmos genéticos e métodos baseados em heurísticas estocásticas em problemas de desenho de grafos foram encontrados em Branke et al. [47], que utilizaram algoritmos genéticos para desenhos de grafos não direcionados. Nesse trabalho, uma implementação paralela foi utilizada para reduzir o tempo de processamento. Barreto e Barbosa [100] aplicaram o algoritmo genético para problemas de desenhos de grafos não direcionados, considerando alguns critérios estéticos. Rosete-Suarez et al. [101] aplicaram a heurística estocástica "hill-climbing" para desenho de grafos. Vrajitoru [102] introduziu a aplicação de otimização multiobjetivo com algoritmos genéticos para problemas de desenho de grafos. Gudenberg et al. [103] desenvolveu e descreveu um algoritmo evolucionário envolvendo layouts em diagramas de classe UML usando métricas específicas para layouts. H. A.D. do Nascimento [104] apresenta um algoritmo genético voltado para desenho de grafos direcionados que utilizam dicas do usuário. Um quadro interactivo é considerado onde os usuários podem se concentrar em regiões do desenho que precisam de melhorias importantes, ou incluir conhecimentos do domínio como restrições de layout. O trabalho descreve como o foco e as restrições dos usuários são gerenciadas pelo algoritmo genético. A combinação das habilidades do usuário com ferramentas automáticas e o uso do algoritmo genético permitem o desenvolvimento de métodos de otimização mais flexíveis e eficientes, quando em comparação aos tradicionais não-interativos. Finalmente, Kunts et al. [96] apresentaram um algoritmo genético híbrido para

minimizar o número de cruzamentos em dígrafos hierárquicos. Não foram encontrados trabalhos relacionados com a utilização de algoritmos genéticos na abordagem de desenho de grafos *topologia-forma-métrica* ou mesmo desenhos ortogonais no grid.

3.7. Parâmetros do Algoritmo Genético

Testes foram efetuados com diversos parâmetros de algoritmo genético de forma a definir quais seriam os parâmetros a serem utilizados no AG para a abordagem híbrida. Foram utilizados 5 grafos com número de vértices variando de 10 a 50 para testes relacionados com a definição do tamanho da população e para os testes de validação das novas metodologias híbridas foram utilizados 11 grafos com número de vértices variando de 10 a 500. São mostrados a seguir os testes realizados, bem como os resultados obtidos com os mesmos:

a) Função de aptidão:

Os cálculos dos valores que compõem a função de aptidão para cada indivíduo levam em consideração os seguintes critérios estéticos: (*i*) o *número de cruzamentos*, representado por f_X; (*ii*) *o número de dobras*, representado por f_B ; (*iii*) a *soma total dos comprimentos das arestas* do grafo, representada por f_L. Todos estes critérios estéticos devem ser minimizados para a obtenção de um desenho de grafos mais otimizado possível, considerando os pesos correspondentes para cada critério estético de acordo com o seu grau de importância. Consequentemente, o cálculo dos valores da função de aptidão é obtido através da seguinte expressão:

$$\phi(S_{t,i}) = \alpha_1 f_X + \alpha_2 f_B + \alpha_3 f_L. \tag{3.22}$$

com $\alpha_i \in [0,1]$. No caso em estudo, foram selecionados os valores $\alpha_1 = 0.5$, $\alpha_2 = 0.3$, e $\alpha_3 = 0.2$ de forma que $\alpha_1 > \alpha_2 > \alpha_3$.

O cálculo de f_X, f_B e f_L requer a execução das etapas de *planarização*, *ortogonalização* e *compactação* para o embutimento planar representado pelo indivíduo $S_{t,i}$. Estas duas etapas são resolvidas também pela heurística clássica (*topologia-forma-métrica*) implementada através da literatura [17-19]. Conforme já citado, é com base neste senso que esta abordagem desenvolvida é considerada uma abordagem híbrida.

b) Número de execuções do algoritmo genético:

O número de execuções do algoritmo genético foi escolhido arbitrariamente como sendo 10 execuções para cada instância de grafo do conjunto de teste.

c) Critério de parada:

Como critério de parada, o algoritmo genético é executado até que a melhor solução encontrada não é melhorada para as últimas M gerações. O valor de M é arbitrariamente escolhido como uma proporção em relação ao número de vértices da instância do grafo tratado. Por exemplo, $M = 5$ para $V \leq 100$ e $M = 20$ para $100 < V \leq 500$.

Para garantir que a escolha do critério de parada baseado no cálculo de M seja considerada uma estratégia interessante para o AG, foram feitos testes considerando alguns números fixos de gerações, o que é mostrado na *Tabela 3.3*:

Tabela 3.3 – Tabela de resultados dos testes para escolha do número de gerações do algoritmo genético na abordagem híbrida.

Caso V-N	f_X	f_B	f_L	***F***	**N. Gerações**
10-30	0	3	23	55	5
	0	3	22	53	10
	0	3	22	53	100
	0	3	22	53	1000
20-30	1	2	48	107	5
	1	1	48	104	10
	1	2	45	101	100
	1	1	48	104	1000
30-30	1	6	91	205	5
	1	5	85	190	10
	1	6	92	207	100
	1	5	85	190	1000
40-30	8	7	126	313	5
	7	6	122	297	10
	4	6	120	278	100
	5	5	121	282	1000
50-30	13	9	238	568	5
	13	13	224	552	10
	14	14	225	562	100
	14	14	224	560	1000
100-30	66	39	298	1043	5
	75	40	291	1077	10
	74	43	290	1079	100
	75	40	291	1077	1000

Os resultados da *Tabela 3.3* mostram que, na maioria das vezes, com 10 gerações, já se havia chegado ao melhor resultado, portanto, a escolha do critério de parada como sendo o valor de *M*, conforme mostrado anteriormente, é suficiente para se garantir a convergência do algoritmo genético para o problema em estudo.

d) Definição do tamanho da população:

A *Tabela 3.4* apresenta resultados para diversas combinações de número de vértices variando de 10 a 50 e o tamanho da população variando de 10 a 50, cujo objetivo é o de avaliar para qual tamanho da população os resultados são mais interessantes. A partir desta análise, os demais testes são realizados para o tamanho da população definido. Nesta tabela, f_X representa o *número de cruzamentos*, f_B o *número de dobra*, f_L a *soma total dos comprimentos das arestas* e *F* é a função de aptidão.

Tabela 3.4 - Resultados para combinações de número de vértices (***V***) e tamanho da população (***N***).

Caso V-N	f_X	f_B	f_L	***F***
10-10	0	3	23	55
10-20	0	3	23	55
10-30	0	3	22	53
10-40	0	3	22	53
10-50	0	3	22	53
20-10	1	2	48	107
20-20	1	1	45	98
20-30	1	1	45	98
20-40	1	1	45	98
20-50	1	1	45	98
30-10	1	5	92	204
30-20	2	5	91	207
30-30	1	5	85	190
30-40	1	5	85	190
30-50	1	5	85	190
40-10	7	8	125	309
40-20	5	5	121	282
40-30	5	5	121	282
40-40	5	5	121	282
40-50	5	5	120	280
50-10	18	13	239	607
50-20	16	13	237	593
50-30	16	13	235	589
50-40	14	13	238	585
50-50	16	13	234	587

De acordo com os resultados mostrados na *Tabela 3.4*, o tamanho da população ideal para os demais testes a serem efetuados é $N = 30$. Este valor foi escolhido porque apresentou resultados mínimos para todos os objetivos analisados, quando comparado aos demais resultados. Para $N > 30$ os resultados se repetiram na maioria dos casos, porém quanto maior for o valor de N, maior será o custo computacional para processar o algoritmo genético, o que justifica a escolha de $N = 30$ como tamanho ideal para a população. Foram efetuadas 10 execuções do AG e escolhidos os resultados com melhores valores para a soma ponderada F.

e) Para definição dos pontos de corte utilizados nos operadores de cruzamento e mutação:

Para ambos os operadores de cruzamento e mutação é necessário escolher o ponto de início do cruzamento e mutação, bem como o ponto fim para os mesmos (também conhecidos como pontos de corte). Normalmente, essa escolha é feita aleatoriamente, mas observamos que o problema tratado possui uma característica especial, que mostra que as alterações na ordenação das arestas a serem inseridas para formar o embutimento planar são mais significativas se o ponto de início estiver em torno da metade do genoma e o ponto de fim estiver no final do genoma. A razão para este comportamento é que as primeiras arestas inseridas dificilmente são responsáveis por cruzamentos, mas, sim, aquelas que são inseridas nas proximidades da metade para o final do genoma. São feitos testes com as escolhas destes pontos de corte de forma pré-definida, ou seja, o ponto de início sendo igual a metade do genoma e o ponto de fim igual ao final do genoma. Foram feitos, também, testes com escolhas aleatórias para estes pontos de início e fim. Observou-se que o número de gerações aumentou significativamente (ver *Tabela 3.5*) para os casos nas quais as escolhas destes pontos são aleatórias. Por esta razão, usando deste conhecimento adquirido sobre o comportamento do problema e visando melhorar o custo computacional do algoritmo genético, decidiu-se usar a distribuição Gaussiana para escolha dos pontos de corte. Os parâmetros desta distribuição são: média $\mu = 0{,}5$, e o desvio-padrão $\sigma = 1 / N$, para selecionar o ponto de início e média $\mu = 1{,}0$ e desvio-padrão $\sigma = 1 / N$, para selecionar o ponto de fim dos operadores genéticos no problema em estudo. N é o número de indivíduos na população. A *Tabela 3.5* mostra as diferenças em termos de número de gerações para cada caso citado. Na tabela, "*pré-definido"* significa que a escolha dos pontos de corte é pré-definida (ponto de início igual ao meio do genoma e ponto de fim igual ao fim do genoma). "*Aleatória"* significa que a escolha dos pontos de corte foi feita de forma aleatória e, finalmente, na *Distribuição Gaussiana* significa que a escolha dos pontos de corte foi feita usando a distribuição gaussiana, com os parâmetros escolhidos conforme mostrado nesta seção. *NG* é o número de gerações obtido.

Tabela 3.5 - Resultados para diferentes escolhas dos pontos de corte.

-	**Pre-definido**	**Aleatório**	**Distribuição Gaussiana**
V	NG	NG	NG
10	7	16	7
20	12	22	15
30	9	17	5
40	12	16	11
50	8	14	9
100	6	12	8

Para a metodologia desenvolvida neste trabalho é utilizada a distribuição gaussiana para a escolha dos pontos de corte dos operadores genéticos.

f) Probabilidade de cruzamento e probabilidade de mutação:

Foram testadas algumas probabilidades, tanto de cruzamento quanto de mutação e ficou definida a utilização das seguintes probabilidades:

- Probabilidade de cruzamento utilizada: 0,8;
- Probabilidade de mutação utilizada: 0,1;

g) Operador de seleção:

A implementação do algoritmo genético emprega seleção por torneio binário;

h) Estratégia para o uso de mais de um operador de cruzamento e de mutação:

São utilizados alguns operadores de cruzamento (PMX e OX) e alguns operadores de mutação (*scramble*, *swap*, inversão e inserção) no processo evolucionário da abordagem desenvolvida. Para tanto, a estratégia utilizada para a escolha de qual operador de cruzamento e mutação utilizar durante o processo evolucionário é a seguinte:

h.1) Para o operador de cruzamento:

rcross = 0; //para checar a taxa de cruzamento
rndc = rand() % 2; //escolha aleatória de qual operador de cruzamento aplicar
Para (cada dois individuos da população)
{

```
        rcross = rand() % N; //escolha um valor aleatoriamente de 0 a N.
        //Se o valor escolhido for menor ou igual à taxa de probabilidade de cruzamento
        Se (rcross <= CP)    //CP = taxa de probabilidade de cruzamento.
        {
                Escolha(rndc)
                {
                        caso 0: cruzamento parcialmente mapeado (PMX)
                                //cruzando os pais selecionados
                        caso 1: cruzamento por ordenação (OX)
                                //cruzando os pais selecionados
                }
        }
        Senão
        {
                copie os pais sem fazer cruzamentos
        }
}
```

h.2) Para o operador de mutação:

```
rmut = 0; //para checar taxa de mutação
rndm = rand() % 4; //escolha aleatória de qual operador de mutação utilizar entre as 4
opções disponíveis (scramble, swap, inserção e inversão)
Para ( toda a população)
{
        rmut = rand() % N; //escolhe um valor aleatório de 0 a N.
        //Se o valor escolhido for menor ou igual à taxa de probabilidade de mutação
        Se (rmut <= MP) //MP = taxa de probabilidade de mutação.
        {
                //aplica o operador de mutação de acordo com o valor de rndm
                Escolha(rndm)
                {
                        caso 0: mutação por SCRAMBLE(SCR)
                        //muta o individuo e coloca-o para compor a nova população
                        caso 1: mutação por SWAP(SWP)
                        //muta o individuo e  coloca-o para compor a nova população
                        caso 2: mutação por INVERSÃO(INV)
                        //muta o individuo e coloca-o para compor a nova população
                        caso 3: mutação por INSERÇÃO(INS)
                        //muta o individuo e coloca-o para compor a nova população
```

```
            }
        }
        senão
            copia o individuo na nova população sem mutar
}
```

Os parâmetros mostrados nesta seção foram utilizados nas implementações do algoritmo genético para as três abordagens híbridas. A primeira que utiliza a *soma ponderada dos objetivos* como função de aptidão doravante é denominada TSM-WS, a segunda que utilizada a *tomada de decisão multicritério em ambiente fuzzy* para ordenação das alternativas é denominada TSM-*FUZZY* e a última que utiliza o algoritmo genético multiobjetivo NSGAII é denominada TSM-NSGAII.

3.8. Abordagem híbrida TSM-WS

O algoritmo desenvolvido para a abordagem híbrida (*topologia-forma-métrica* + *AG*) usando a soma ponderada dos objetivos como função de aptidão no processo evolucionário é mostrado a seguir:

Algoritmo Hibrido (*TSM-WS-GA*)

Entrad*a:* grafo *G;*

Saida: desenho ortogonal otimizado;

1 - Geração da População inicial:

N = tamanho da população*;*

a) Gere aleatoriamente a ordenação de inserção das arestas de G (representada pela permutação de inteiros).

2 – Calculo da Aptidão:

i = 0;

Enquanto (*i* < *N* **) faça**

{

- Submeta *i* à etapa de planarização da *topologia-forma-métrica* para obtenção do embutimento planar (Γ_i) de *i*, bem como o número de cruzamentos (fx_i) de *i*, o qual irá compor a função aptidão;

- Submeta o embutimento planar (Γ_i) a etapa de ortogonalização da *topologia-forma-métrica* para obtenção da representação ortogonal ***H***, bem como o número de dobras (f_{Bi}) para compor a função aptidão;
- Submeta a representação ortogonal ***H*** a etapa de compactação da *topologia-forma-métrica* para obtenção da soma total das arestas (f_{Li}) que comporá a função aptidão;
- Calcule o valor da função aptidão para o indivíduo *i* de acordo com a equação ($F_i = 0.5 \bullet f_{Xi} + 0.3 \bullet f_{Bi} + 0.2 \bullet f_{Li}$);
- $i = i + 1$;

}

3 - Aplique os operadores do algoritmo genético para gerar a nova população;

4 – Volte ao passo 2 até que o critério de parada seja atingido;

Os resultados obtidos pela abordagem que utiliza a soma ponderada como função de aptidão no processo evolucionário serão mostrados na seção de resultados (*Seção 3.11*).

Na próxima seção são introduzidos os conceitos relacionados com *tomada de decisão multicritério em ambiente fuzzy*, pois os mesmos são aplicados neste trabalho no processo evolucionário do AG, em substituição a soma ponderada. Desta maneira é garantida a escolha de soluções mais harmoniosas entre o conjunto de soluções encontradas pela metodologia desenvolvida.

3.9. Abordagem híbrida *fuzzy* TSM-*FUZZY*

Os processos que envolvem problemas de tomada de decisão e que dependem, na maioria das vezes, do decisor humano estão relacionados com a percepção deste decisor em torno das alternativas que estão sendo comparadas. Assim, incertezas, imprecisões e ambiguidades são aspectos intrínsecos à maioria dos problemas do mundo real que envolvem tomada de decisão. Uma questão importante que deve ser considerada no uso da tomada de decisão multicritério é a qualidade da solução em si. Esta qualidade é considerada alta se os níveis de satisfação dos objetivos são iguais ou próximos uns dos outros (que dão origem às chamadas soluções harmoniosas), quando os níveis de importância das funções objetivo são iguais. Não é difícil estender este conceito para os casos cujos níveis de importância das funções objetivo são diferentes: as soluções devem ser harmoniosas, levando em consideração os fatores de importância que correspondem a cada objetivo. Deste ponto de vista, deve ser registrada a validade e conveniência da direção relacionada com o princípio do resultado garantido [105]. Outras direções na

tomada de decisão multicritério, em especial aquelas indicadas acima, podem levar a soluções com elevados níveis de satisfação de alguns dos critérios e, ao mesmo tempo, assegurar baixos níveis de satisfação dos demais critérios. Esta situação pode ser totalmente inaceitável do ponto de vista de qualidade de soluções (por exemplo, [105], [106], [107] e [108]).

Uma possível maneira de se tratar estas incertezas nos modelos de tomada de decisão é através da utilização dos conceitos relacionados com os conjuntos *fuzzy*, pois a lógica *fuzzy* mostra-se como solução natural para a criação de modelos reais e flexíveis os quais são capazes de representar as incertezas do julgamento humano.

A metodologia que criou o conceito de soluções harmoniosas em problemas de tomada de decisão multicritério foi desenvolvida e seus resultados foram apresentados em [109], [110] e [111], com base na abordagem de Bellman-Zadeh para tomada de decisões em ambiente *fuzzy* [112] e foi largamente utilizada neste trabalho.

Nesta abordagem. a preparação das informações para a tomada de decisão deve ser realizada em três etapas:

- Substituição dos objetivos (que correspondem, por exemplo, a "*número de cruzamentos"*, "*número de dobras"*, "*soma total dos comprimentos das arestas*", etc.) por um conjunto *fuzzy* (através da função de pertinência);
- Aplicação de metodologia para agregação dos critérios;
- Aplicação de metodologia para ordenação das alternativas de solução.

A substituição dos valores dos objetivos por um conjunto *fuzzy* será mostrada nos próximos parágrafos. Como operador de agregação dos critérios, a abordagem utiliza o operador *min* e para sua ordenação utiliza o operador *max*.

Quando aplicada a abordagem de Bellman-Zadeh [112] para tomada de decisão em ambiente *fuzzy* com o objetivo de resolver problemas multicritérios, cada função objetivo $f_i(x)$ pode ser substituída por uma função objetivo *fuzzy* ou um conjunto *fuzzy*:

$$F_i(x) = \{x,\ \mu_{F_i}(x)\},\ x \in L \qquad (3.23)$$

Em que *i* representa a função objetivo em análise e *L* é um conjunto das soluções viáveis no problema de desenho de grafos.

A solução *fuzzy* $D(x)$, com base nos conjuntos *fuzzy* $F_i(x)$, que devem refletir a qualidade dos *m* correspondentes indicadores, é formada como resultado da interseção

$$D(x) = \bigcap_{i=1}^{m} F_i(x) \tag{3.24}$$

com a função de pertinência

$$\mu_D(x) = \bigwedge_{i=1}^{m} \mu_{F_i}(x) = \min_{i=1,2,..m} \mu_{F_i}(x) \tag{3.25}$$

A sua utilização permite obter uma solução que prove o grau máximo de pertinência para a solução *fuzzy* $D(x)$

$$\max \mu_D(x) = \max_{x \in L} \min_{i=1,2,..,m} \mu_{F\,i}(x) \tag{3.26}$$

Do ponto de vista formal, o problema da escolha da alternativa multicritério é reduzido à busca por

$$x^* = \arg\max_{x \in L} \min_{i=1,2,...,m} \mu_{F_i}(x). \tag{3.27}$$

Para a obtenção de (3.27), é necessário construir funções de pertinência $\mu_{F_i}(x)$, $i = 1,2,...,m$ que devem refletir o grau de qualidade do seu próprio ótimo para $f_i(x),\ i = 1,2,...,m$. Para se atingir esta meta [109], [110] e [113], pode-se aplicar a função de pertinência:

$$\mu_{F_i}(x) = \left[\frac{f_i(x) \quad - \min_{x \in L} f_i(x)}{\max_{x \in L} f_i(x) - \min_{x \in L} f_i(x)} \right]^{\lambda_i}, \tag{3.28}$$

para funções objetivo que demandam sua maximização ou pela utilização de funções de pertinência:

$$\mu_{F_i}(x) = \left[\frac{\max_{x \in L} f_i(x) \ - \quad f_i(x)}{\max_{x \in L} f_i(x) - \min_{x \in L} f_i(x)} \right]^{\lambda_i}. \tag{3.29}$$

para funções objetivo que demandam sua minimização.

Em (3.28) e (3.29), λ_i i = 1, . . . ,*m* são fatores de importância para as correspondentes funções objetivo (critérios estéticos), quando distinguimos a importância dos objetivos.

Levando em consideração o que foi descrito, é necessário indicar que se pode utilizar (3.25) como função de aptidão guiando o processo evolucionário do algoritmo genético na busca por soluções mais harmoniosas no problema de desenho de grafos. Desta maneira, a abordagem descrita é aplicada no processo evolucionário do AG para a seleção dos indivíduos que garantirão soluções harmoniosas para as próximas gerações. Além disto, esta abordagem é também aplicada para a escolha do indivíduo mais harmonioso ao final da geração de todos os resultados encontrados pelo algoritmo genético, considerando as 10 execuções do AG. Assim, são escolhidos aqueles que, do ponto de vista de todos os critérios estéticos considerados, apresente maior harmonia entre os critérios estéticos. Com isto, é possível escolher o melhor resultado para cada grafo baseado no conceito de soluções harmoniosas e, desta forma, garantir que a escolha das alternativas, feita com base nesta abordagem, leve em consideração todos o objetivos tratados no problema, o grau de importância de cada um deles (quando necessário), bem como as incertezas inerentes ao tratamento de problemas desta natureza. Na nossa abordagem o grau de importância (λ_i) considerado foi: para $f_X \rightarrow \lambda_1 = 0.5$, $f_B \rightarrow \lambda_2 = 0.3$ e $f_L \rightarrow \lambda_3 = 0.2$.

O algoritmo desenvolvido para a abordagem híbrida (*topologia-forma-métrica + AG*), usando o modelo *fuzzy na tomada de decisão multicritério* no processo evolucionário, é mostrado a seguir:

Algoritmo Hibrido (*TSM-*FUZZY*-FDM*)

Entrada: grafo *G;*

Saida: desenho ortogonal otimizado;

1 - Geração da População inicial:

N = tamanho da população de indivíduos*;*

a) Gere aleatoriamente a ordenação de inserção das arestas de G (representada pela permutação de inteiros).

2 – Cálculo da Aptidão:

i = 0;

Enquanto (*i* < *N* **) faça**

{

- Submeta a solução *i* a etapa de planarização da *topologia-forma-métrica* para obtenção do embutimento planar (Γ_i) de *i* bem como o número de cruzamentos (fx_i) de *i*;
- Submeta o embutimento planar (Γ_i) à etapa de ortogonalização da *topologia-forma-métrica* para obtenção da representação ortogonal ***H***, bem como o número de dobras (f_{Bi});
- Submeta a representação ortogonal ***H*** à etapa de compactação da *topologia-forma-métrica* para obtenção da soma total das arestas (f_{Li});
- Calcule o valor das pertinencias *fuzzy* μ_{F_X} , μ_{F_B} e μ_{F_L} ;
- Calcule a agregação *max-min fuzzy* μ_D ;
- *i = i + 1*;

}

3- Grave o melhor indivíduo de acordo com o valor de sua função de aptidão;

4- Aplique os operadores do algoritmo genético para gerar a nova população. Cada operador de cruzamento (PMX ou OX), para produzir os descendentes, é selecionado com probabilidade igual a 0,50. Os operadores de mutação utilizados (*scramble, swap, insert e invert*) para produzir cada descendente são também selecionados aleatoriamente com chances iguais de 0,25.;

5- Aplique o operador de seleção;

6- Volte ao passo 2 até que o critério de parada seja atingido.

Na seção de resultados (*Seção 3.11*) são mostrados os resultados obtidos com esta abordagem.

Na próxima seção serão introduzidos os conceitos relacionados ao algoritmo genético multiobjetivo (NSGAII), pois os mesmos são aplicados neste trabalho no processo evolucionário do AG.

3.10. Abordagem híbrida TSM-NSGAII

Primeiramente serão apresentados os conceitos que envolvem o algoritmo genético NSGAII e na sequência a abordagem híbrida TSM-NSGAII será apresentada.

Diversos algoritmos genéticos multiobjetivos foram desenvolvidos [114], [115], [116], [117], [118], [119], [120], [121], [122], [123] e aplicados em problemas de diversas naturezas. Dentre eles, o NSGAII é um algoritmo genético multiobjetivo [124] que consiste em um ordenamento elitista por não-dominância da população. Ele trabalha com a população pai P_t para gerar a população filha Q_t, como acontece nos AG's convencionais. Na primeira iteração, gera-se uma população P_1, e os seus descendentes Q_1, as quais são ordenadas por não-dominância. Cada solução é separada por nível de não-dominância, denominada por frente de Pareto (F_i). A primeira frente de Pareto é completamente não-dominada e a segunda é dominada por indivíduos da primeira frente e assim sucessivamente. A cada solução é atribuída um valor de aptidão de acordo com a frente em que ocupa. Por exemplo, para a solução que compõe a primeira frente de Pareto (F_1) é atribuído o valor 1, para a solução que compõe a segunda frente (F_2) é atribuído o valor 2 e assim sucessivamente. Além do valor de aptidão atribuído a cada indivíduo, é calculado e atribuído também um parâmetro chamado de *crowding distance* (distância de multidão). Para o cálculo deste parâmetro, leva-se em consideração o quão perto o indivíduo está de seus vizinhos. Uma maior média de *crowding distance* resultará em uma melhor diversidade na população. Os pais são selecionados na população através da seleção por torneio binário, baseado no ranqueamento por aptidão e *crowding distance.* Um indivíduo é selecionado se seu valor de aptidão no ranqueamento é menor que outros e, caso sejam iguais, é avaliado se sua *crowding distance* é maior que a do outro. Os pais selecionados geram descendentes através da aplicação dos operadores de cruzamento e mutação adequados, gerando, assim, a população filha Q_1. Tanto P_t como Q_t são de tamanho N [41], [121].

As populações P_t e Q_t são unidas em uma população $R_t = P_t \cup Q_t$, com $|R| = 2N$. Para as seguintes gerações $t = 1,2,...$, o algoritmo NSGAII trabalha com a população R_t.

Realiza-se um ordenamento por não-dominância sobre R_t, obtendo as fronteiras F_1, F_2,...F_n e todos estes conjuntos são inseridos na nova população P_{t+1}. Dado que apenas N soluções podem ser inseridas, N soluções de R_t são descartadas. Para preencher as P_{t+1} se começa pelas soluções em F_1, depois em F_2 e assim sucessivamente. Cada conjunto F_i deve ser inserido na sua totalidade em P_{t+1}, isto acontece enquanto $|P_{t+1}| + |F_i| \leq N$. Ao inserir um F_j tal que $|F_j| > N - |P_{t+1}|$, o algoritmo NSGAII escolhe aquelas soluções de F_j que estejam melhor espalhadas (métrica de *crowding distance*). A *Figura 3.26* ilustra uma iteração para o NSGAII.

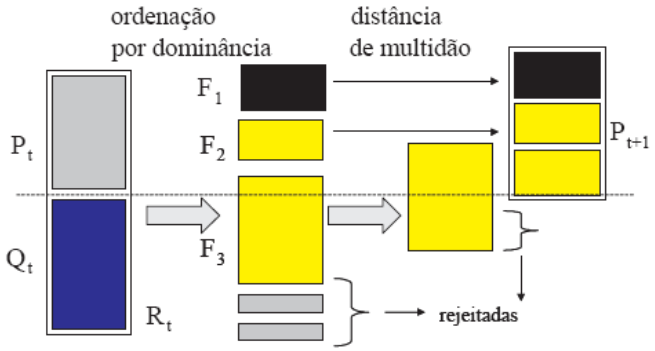


Figura 3.26 –Esquema do modelo do NSGAII [121].

O algoritmo NSGAII introduz o método chamado de distância de multidão (*crowding distance*). Nele as distâncias de cada indivíduo em relação à sua vizinhança são calculadas. Os conjuntos F_j são ordenados decrescentemente em relação às suas distâncias de multidão, então se copiam as primeiras $N - |P_{t+1}|$ soluções de F_j para P_{t+1}.

A distância de multidão d_i de uma solução i representa uma estimativa do perímetro formado pelo cubóide cujos vértices são os seus vizinhos mais próximos. A *Figura 3.27* mostra a distância de multidão para a solução i . Quanto maior for o cubóide de i, mais afastada encontra-se i dos seus vizinhos. As soluções extremas em cada objetivo terão um cubóide infinito.

Finalmente, gera-se uma nova população $|Q_{t+1}|$ a partir de $|P_{t+1}|$ usando os operadores de seleção por torneio, com base nos valores de distância de multidão, bem como os operadores de cruzamento e mutação até que o critério de parada seja atingido.

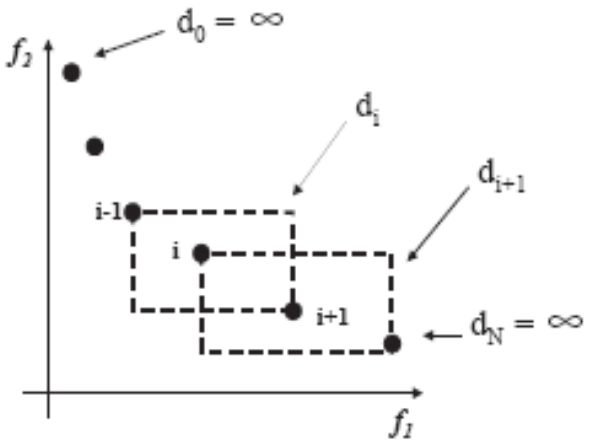


Figura 3.27 –Cuboides para cálculo da distância de multidão no NSGAII [121].

O passo a passo para se encontrar a distância de multidão no NSGAII é descrito no algoritmo a seguir:

***Algoritmo* Distância de Multidão (NSGAII)**

Inicio

F_i : Conjunto de soluções na fronteira i
l denota o número de soluções em F_i.
1: **Para** cada solução em F_i **atribui-se** $d_i = 0$.
2: **Para** cada função objetivo $m = 1, 2, ..., M$.
3: **Ordena-se** em ordem decrescente as soluções por f_m na lista I^m.
4: **Para** cada solução extrema (mínimo e máximo) em cada um dos m objetivos
5: **Faz-se** $d_{I_1^m} = d_{I_l^m} = \infty$.
6: **Para** as soluções $i = 2, ..., l$ -1, **calcula-se**:

$$d_{I_i^m} = d_{I_i^m} + \frac{f_m^{(I_{i+1}^m)} - f_m^{(I_{i-1}^m)}}{f_m^{\max} - f_m^{\min}} \tag{3.30}$$

Em que :

I_i^m representa a i-ésima solução na lista ordenada pelo objetivo m. I_1^m e I_l^m são os elementos da lista com menor e maior valor em um objetivo m. $f_m^{I_{i+1}^m}$ e $f_m^{I_{i-1}^m}$ são os valores dos vizinhos i na m-ésima função objetivo. $f_m^{\max}$ e $f_m^{\min}$ são parâmetros dos limites máximos e mínimos em cada objetivo. A *Equação 3.30* assegura que as soluções mais afastadas tenham d_i maior.

***Algoritmo* NSGAII**

Inicio

P : População pai.
Q : População filha.
N : Tamanho fixo para P e Q.
F_i : Conjunto de soluções na fronteira i.
$nMax$: Máximo de gerações para o algoritmo.
n : Número de geração atual.
1: Gerar a população inicial P_0 e Q_0.
2: Atribuir $t = 0$. //tempo inicial igual a zero.
3: Realizar a seleção, o cruzamento e a mutação para gerar a população filha Q_0.
4: Fazer $R_t = P_t \cup Q_t$.
5: Realizar a ordenação por não-dominância em R_t
6: Criar $P_{t+1} = 0$.

7: **Enquanto** $|P_{t+1} + F_i| < N$.
8: Copiar as soluções de F_i em P_{t+1}.
9: Calcular as distâncias de multidão em F_j (ultima frente de Pareto que não foi possível acomodar na população de tamanho N).
10: Ordenar F_j conforme as distâncias d_j.
11: Copiar as primeiras $N - |P_{t+1}|$ soluções de F_j para P_{t+1}.
12 **Fim-enquanto**
13: Aplicar os operadores de seleção por torneio com base na distância de multidão para os indivíduos de P_{t+1}.
14: Aplicar os operadores de cruzamento e mutação apropriados e gerar a nova população Q_{t+1}.
16: **Se** $n > nMax$ **então** pare
17: **Senão** atribuir $n = n+1$ e voltar ao passo 4.

A complexidade computacional de tempo do algoritmo NSGAII é de $O(MN^2)$ por iteração, onde M é o número de objetivos e N o tamanho da população [121].

A seguir discutiremos os procedimentos adotados para aplicação do algoritmo genético multiobjetivo NSGAII na abordagem híbrida, puramente multiobjetivo, desenvolvida.

Trata-se da abordagem híbrida envolvendo a *topologia-forma-métrica* e o algoritmo genético multiobjetivo *NSGAII*.

Neste caso, o problema é formulado como um problema de otimização combinatória multiobjetivo baseado em permutações de inteiros. As soluções no conjunto de Pareto representam diferentes desenhos ortogonais que podem ser selecionados pelo usuário em tempo real de acordo com sua preferência. Outra possibilidade é fazer esta escolha com base na abordagem de tomada de decisão multicritério em ambiente *fuzzy*, que garante soluções mais harmoniosas.

O algoritmo relacionado com esta abordagem híbrida (*topologia-forma-métrica* + NSGAII) foi desenvolvido e é mostrado a seguir:

Algoritmo Hibrido (*TSM-NSGA-II*)

Entrad*a:* grafo *G;*
Saida: desenho ortogonal otimizado;
1 - Geração da População inicial:

N = tamanho da população*;*

b) Gere aleatoriamente a ordenação de inserção das arestas de G (representada pela permutação de inteiros).

2 – Calculo dos valores dos objetivos:

i = 0;

Enquanto (*i* < *N* **) faça**

{

- Submeta o indivíduo *i* a etapa de planarização da *topologia-forma-métrica* para obtenção do embutimento planar (Γ_i) de *i* e o número de cruzamentos $fx_{(i)}$;
- Submeta o embutimento planar (Γ_i) a etapa de ortogonalização da *topologia-forma-métrica* para obtenção da representação ortogonal ***H***, e o número de dobras $f_{B(i)}$;
- Submeta a representação ortogonal ***H*** com o seu número de dobras, a etapa de *compactação* para obtenção da soma total das arestas $f_{L(i)}$ bem como do desenho final;
- *i = i + 1*;

}

3- Ordene os indivíduos de acordo com a sua *não-dominância* em frentes de Pareto (F_1 a F_n). Tal que: Em *F1* estão os indivíduos totalmente não-dominados, em *F2* estão os indivíduos dominados somente por indivíduos que estão contidos em *F1* e assim sucessivamente;

4- Calcule a distância de multidão (*crowding distance*) e os valores de aptidão para cada indivíduo da população;

5- Aplique os operadores genéticos para geração da população de descendentes. Cada operador de cruzamento (PMX ou OX) que produz os descendentes é selecionado com probabilidade igual a 0,50. O Operador de mutação a ser utilizado (*scramble, swap, insert e invert*) para cada descendente é também selecionado com probabilidade igual a 0.25;

6- Aplique o operador de sobrevivência por elitismo em $P_t \cup Q_t$ para obter P_{t+1};

7- Se o critério de parada não for atingido, volte ao passo 2;

8- Ao atingir o critério de parada, calcule os valores de *agregação fuzzy max-min* na frente de Pareto F_1 final. Esta metodologia é aplicada a fim de se escolher, entre os resultados finais, aquele que represente uma maior hamoniosidade entre todos os resultados pertencentes a F_1. Este será considerado o resultado final (que garante uma maior harmonia) de ocordo com a abordagem de *tomada de decisão multicritério em ambiente fuzzy.*

A principal contribuição desta metodologia é a utilização de um algoritmo genético multiobjetivo para selecionar um número maior de embutimentos planares que possam levar a um desenho final mais otimizado. Em muitas aplicações práticas que envolvem a geração de desenho de grafos otimizados, o custo em termos de tempo gasto e de esforço computacional é elevado. Usando a abordagem multiobjetivo, pode-se procurar por um conjunto de desenhos (representados por vértices e arestas previamente armazenado em um conjunto de dados) e torná-lo disponível para o usuário. Uma vez que as preferências ou grau de importância sobre os critérios estéticos são subjetivos, o usuário pode selecionar e avaliar os desenhos contidos no conjunto de Pareto em tempo real, sem a necessidade de gerar os grafos otimizados novamente. Isso mostra a vantagem de usar uma abordagem multiobjetivo para o desenho automático de grafos.

Nos resultados obtidos foram aplicados a metodologia de tomada de decisão multicritério, em um ambiente *fuzzy*, para encontrar uma solução mais harmoniosa no conjunto solução de Pareto, considerando os três objetivos estéticos mostrados. No entanto, esta metodologia permite fazer uma ordenação das alternativas de acordo com o seu grau de pertinência no conjunto de soluções harmoniosas e, desta forma, permitir ao usuário a escolha das soluções de acordo com o seu grau de harmonia no conjunto de Pareto. Além disto, para o caso onde foi utilizado o algoritmo genético mono-objetivo e a soma ponderada dos objetivos, a principal dificuldade encontrada no uso da soma ponderada está relacionada à seleção do pesos que compõem esta soma ponderada. Esta é considerada uma tarefa não-trivial. Além disto, se o usuário não está satisfeito com o resultado final, é necessário mudar os pesos e executar o processo de otimização novamente. Por este motivo, é importante o desenvolvimento de uma metodologia que permita encontrar simultaneamente um conjunto solução, deixando a decisão final para o usuário, de forma a posteriori.

Na seção seguinte são mostrados os resultados obtidos com a abordagem clássica e as abordagens híbridas.

3.11. Resultados das abordagens clássica e híbridas

Para a geração dos casos de teste, seguiu-se o procedimento descrito a seguir:

1. Foram gerados grafos esparsos variando o número de vértices *V*, de 10 - 500 vértices, com a restrição de que o grau de cada vértice não seja superior a 4, uma vez que estamos tratando de desenhos ortogonais. O número de arestas no grafo é gerado de forma a garantir a esparsidade do grafo.
2. Para o caso clássico, aquele onde se obtém-se o embutimento planar fixo na *topologia-forma-métrica*, as arestas são inseridas na sequência de sua identificação,

em ordem crescente. Este embutimento planar é submetido aos algoritmos de *ortogonalização* e *compactação* da abordagem clássica *topologia-forma-métrica*. Desta forma, obtém-se o resultado da função de aptidão com base na abordagem clássica *topologia-forma-métrica*. Neste caso, sem o uso do algoritmo genético.

3. Cria-se a população inicial a partir da sequência clássica (ordenação pelos identificadores das arestas na ordem crescente) de inserção das arestas, fazendo a permutação aleatória da sequência de arestas a serem inseridas. Desta maneira, são feitas quantas permutações forem necessárias para formar o número *N* de indivíduos na população. Obtém-se, assim, a população inicial. Cada indivíduo da população inicial é submetido às etapas de *planarização, ortogonalização* e *compactação,* de forma a obter-se os valores para cada critério estético: f_X, f_B e f_L e da função de aptidão *F*.
4. Executa-se o algoritmo genético na população inicial, gerando então a nova população. Cada novo indivíduo (embutimento planar) da nova população é submetido também às etapas de *planarização*, *ortogonalização* e *compactação* de forma a obter-se os valores para cada critério estético: f_X, f_B e f_L e da função de aptidão *F*.

O processo evolucionário do AG é tratado de acordo com cada metodologia desenvolvida e testada. Os resultados, tanto para a abordagem clássica *topologia-forma-métrica,* como para as abordagens desenvolvidas, serão mostrados nas tabelas nas próximas seções. Foi feita a análise também para o caso em que se efetua a permutação aleatória nos embutimentos planares e os submetem às etapas de *planarização*, *ortogonalização* e *compactação,* sem a aplicação do algoritmo genético. Seus resultados também serão mostrados.

Os resultados são tabelados para cada caso: clássico (TSM), aleatório e sem o uso do AG, para cada abordagens híbridas TSM-WS, TSM-*FUZZY* e TSM-NSGAII individualmente e também são tabelados todos os resultados em uma única tabela na *Seção 3.12*, *Tabela 3.11* para facilitar a análise conjunta dos mesmos. Na maioria dos casos foi necessário suprimir a coluna de resultado para a função *F*, na tabela única, por questões de espaço. A tabela de resultados individuais, para cada caso foi mantida por trazer os resultados completos.

3.11.1. Resultados da abordagem clássica

Nesta seção foram gerados os resultados considerando apenas a abordagem clássica *topologia-forma-métrica*. Estes resultados são fundamentais, pois a análise comparativa de resultados entre a abordagem clássica e as abordagens desenvolvidas é utilizada para validação destas últimas. Seus resultados são mostrados na *Tabela 3.6*.

Nesta tabela e para todas as tabelas onde contém estas informações, *V* é o número de vertices do grafo, f_X é o *número de cruzamentos* obtidos, f_B é o *número de dobras*, f_L é a *soma total dos comprimentos das arestas* do grafo, e finalmente, *F* é o valor da função de aptidão dada pela *Equação 3.22* e obtida após a computação dos passos de *planarização*, *ortogonalização* e *compactação*.

Tabela 3.6 - Resultados para a abordagem clássica.

V	f_X	f_B	f_L	F
10	0	3	23	55
20	1	2	55	121
30	1	5	93	206
40	6	6	164	376
50	24	19	352	881
100	119	49	528	1798
150	137	59	632	2126
180	194	75	781	2757
200	509	172	854	4769
250	618	197	1144	5969
500	608	173	2748	9055

Na próxima seção serão mostrados os resultados sem a utilização do algoritmo genético. Neste caso, as permutações são geradas aleatoriamente e cada embutimento planar é submetido às etapas de *planarização*, *ortogonalização* e *compactação,* para o cálculo da soma ponderada dos objetivos.

3.11.2. Resultados sem o uso do AG.

Os resultados gerados considerando somente a variação da ordem de inserção das arestas sem o uso do algoritmo genético são mostrados na *Tabela 3.7*.

Tabela 3.7 - Resultados utilizando ordenação aleatória das arestas sem o uso do algoritmo genético.

Caso V-N	*Estatisticas*	f_X	f_B	f_L	F
10-30	**Melhor**	**0**	**3**	**23**	**55**
	Média	0	3	24	57
	Desvio padrão	0.00	0.00	0.00	0.00
20-30	**Melhor**	**2**	**1**	**51**	**115**
	Média	1	2	55	121
	Desvio padrão	0.47	0.40	0.93	0.90
30-30	**Melhor**	**1**	**5**	**91**	**202**
	Média	1	5	93	208
	Desvio padrão	0.40	0.47	2.68	3.52

40-30	**Melhor**	**8**	**12**	**149**	**374**
	Média	8	11	154	390
	Desvio padrão	0.76	1.59	2.75	0.86
50-30	**Melhor**	**17**	**19**	**350**	**842**
	Média	19	17	359	854
	Desvio padrão	2.43	2.80	4.73	2.24
100-30	**Melhor**	**82**	**44**	**408**	**1358**
	Média	83	45	410	1380
	Desvio padrão	6.92	1.57	26.03	49.91
150-30	**Melhor**	**119**	**51**	**436**	**1620**
	Média	110	50	489	1678
	Desvio padrão	7.01	2.11	30.75	43.11
180-30	**Melhor**	**129**	**65**	**632**	**2104**
	Média	130	65	632	2109
	Desvio padrão	2.79	2.86	1.59	4.74
200-30	**Melhor**	**382**	**138**	**893**	**4110**
	Média	423	155	768	4117
	Desvio padrão	41.75	21.98	135.93	4.89
250-30	**Melhor**	**495**	**159**	**1175**	**5302**
	Média	497	170	1158	5309
	Desvio padrão	2.87	9.92	13.68	6.84
500-30	**Melhor**	**658**	**192**	**2148**	**8162**
	Média	609	201	2259	8168
	Desvio padrão	84.59	15.83	190.65	3.91

Pode-se claramente observar que estes resultados mostrados são resultados intermediários, entre os resultados clássicos (*topologia-forma-métrica*) e os resultados utilizando o AG (ver *Tabela 3.8*), como era esperado. Neste caso, o objetivo é mostrar a superioridade do AG na solução do problema de desenho de grafos.

3.11.3. Resultados da abordagem híbrida TSM-WS

Nesta seção foram gerados os resultados considerando a abordagem híbrida, na qual é utilizada a soma ponderada dos objetivos no processo evolucionário do AG. A *Tabela 3.8* apresenta os resultados para algoritmo genético para cada grafo de teste, identificado pelo número de vértices *V*.

Tabela 3.8 - Resultados para o algoritmo genético usando a soma ponderada como função de aptidão.

Caso V-N	***Estatisticas***	f_X	f_B	f_L	***F***
10-30	**Melhor**	**0**	**3**	**22**	**53**
	Média	0	3	23	55
	Desvio padrão	0.00	0.00	0.53	1.05
20-30	**Melhor**	**1**	**2**	**45**	**101**
	Média	1	1	47	**103**
	Desvio padrão	0.00	0.42	0.92	1.33
30-30	**Melhor**	**1**	**5**	**85**	**190**
	Média	1	5	87	196
	Desvio padrão	0.40	0.00	1.81	4.36
40-30	**Melhor**	**4**	**6**	**120**	**278**
	Média	6	6	123	291
	Desvio padrão	1.35	1.20	4.40	4.24
50-30	**Melhor**	**13**	**13**	**224**	**552**
	Média	14	12	234	573
	Desvio padrão	2.21	2.44	6.83	13.57
100-30	**Melhor**	**66**	**36**	**290**	**1018**
	Média	70	37	318	1102
	Desvio padrão	6.79	2.57	18.96	27.82
150-30	**Melhor**	**86**	**42**	**322**	**1200**
	Média	95	44	372	1351
	Desvio padrão	10.34	4.83	36.96	80.13
180-30	**Melhor**	**101**	**44**	**450**	**1537**
	Média	120	55	461	1689
	Desvio padrão	12.25	8.02	30.36	79.61
200-30	**Melhor**	**330**	**135**	**655**	**3365**
	Média	353	134	656	3477
	Desvio padrão	21.75	5.05	52.98	110.63
250-30	**Melhor**	**436**	**145**	**721**	**4057**
	Média	426	145	846	4256
	Desvio padrão	31.70	7.75	121.28	184.69
500-30	**Melhor**	**317**	**102**	**1351**	**4593**
	Média	320	104	1371	4654
	Desvio padrão	5.31	2.42	24.50	81.79

O gráfico de tempo computacional para o algoritmo genético híbrido, por geração, em relação ao número de vértices, utilizando a soma ponderada dos objetivos como função de aptidão é mostrado na *Figura 3.28* em escala logarítmica em ambos os eixos.

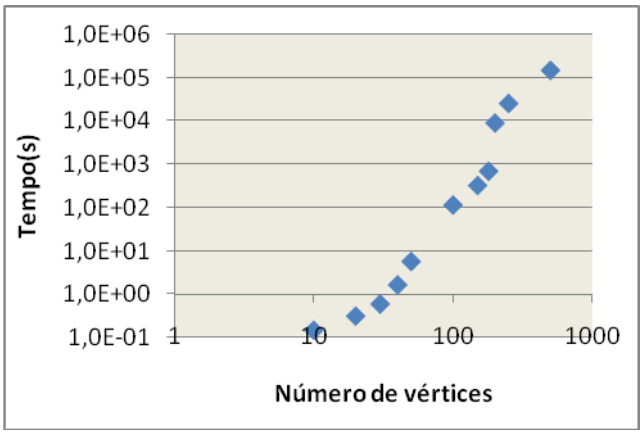


Figura 3.28 – Tempo computacional para o algoritmo genético híbrido por geração em relação ao número de vértices no grafo utilizando a soma ponderada dos objetivos como função de aptidão.

Para as três abordagens híbridas, os tempos computacionais são muito próximos entre si, por esta razão, somente mostrados tempos computacionais para o caso relativo a *Figura* 3.28.

As *Figuras 3.29 a 3.31* mostram os resultados, em termos de desenhos, para alguns dos casos testados na abordagem que utiliza a soma ponderada dos objetivos, como função de aptidão. As figuras estão sempre acompanhadas dos desenhos correspondentes na abordagem clássica (*topologia-forma-métrica*), para fins comparativos. Estas figuras mostram os desenhos obtidos para $V = 20$, $V = 50$ e $V = 100$. Pode-se observar que os desenhos finais gerados pelo algoritmo genético, de fato, apresentam muito mais qualidade e visibilidade, apresentando menos cruzamentos e dobras. Estes resultados ilustram o benefício da utilização da abordagem híbrida, utilizando a soma ponderada como função de aptidão, desenvolvida neste trabalho para desenhos ortogonais no *grid*. Estes resultados foram apresentados no congresso internacional GECCO2010 e publicados em [125].

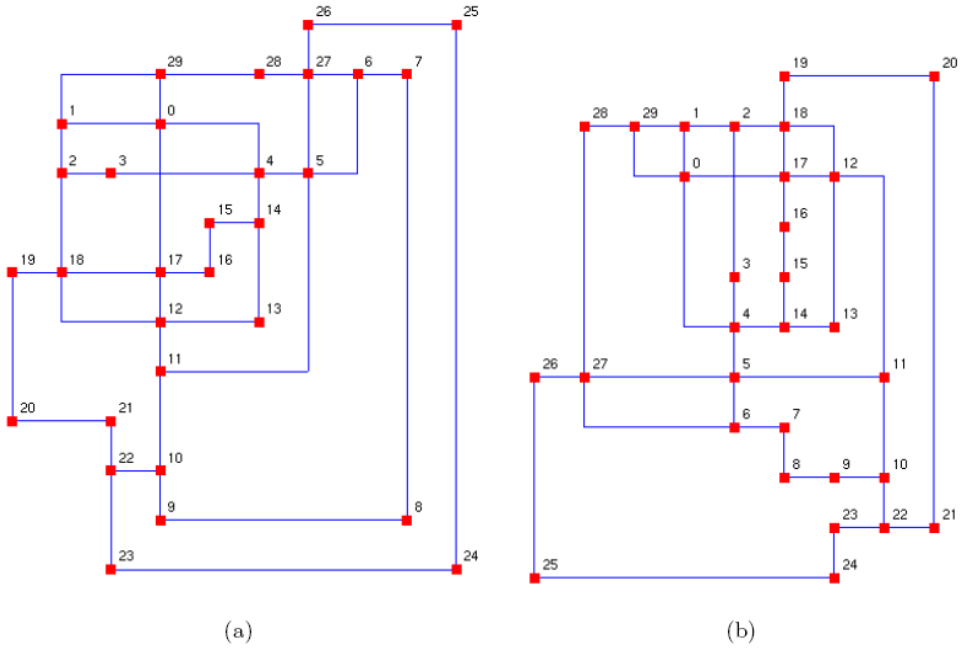


Figura 3.29 – Desenho final para o grafo com V = 30: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético utilizando a soma ponderada como função aptidão.

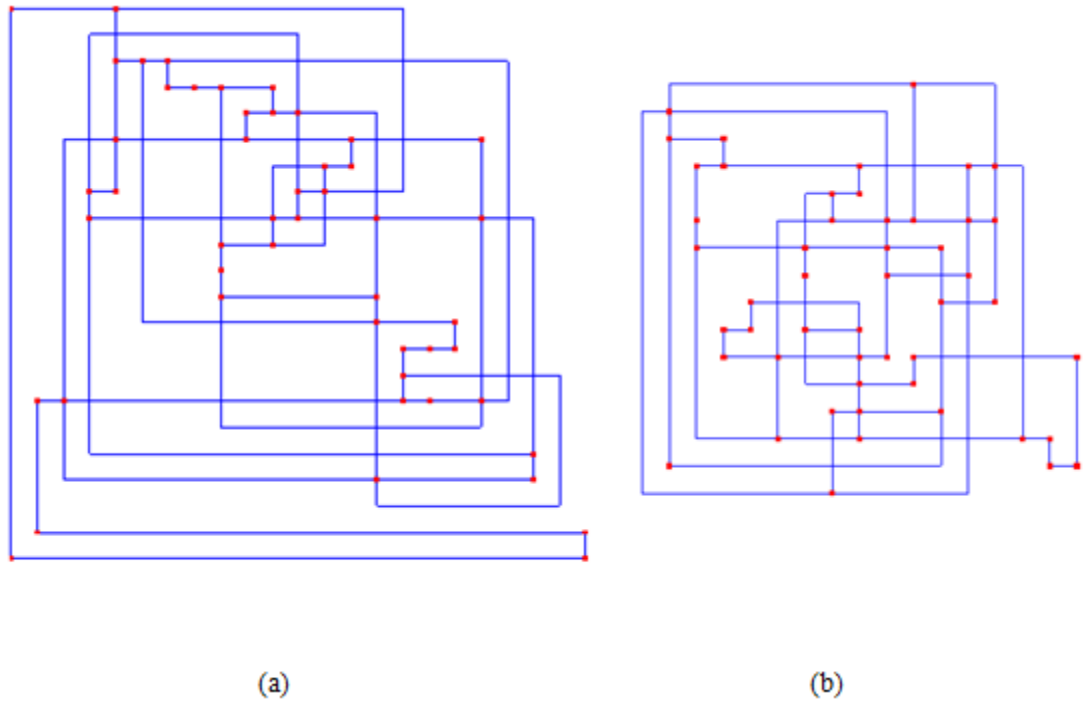


Figura 3.30 – Desenho final para o grafo com V = 50: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético utilizando a soma ponderada como função aptidão.

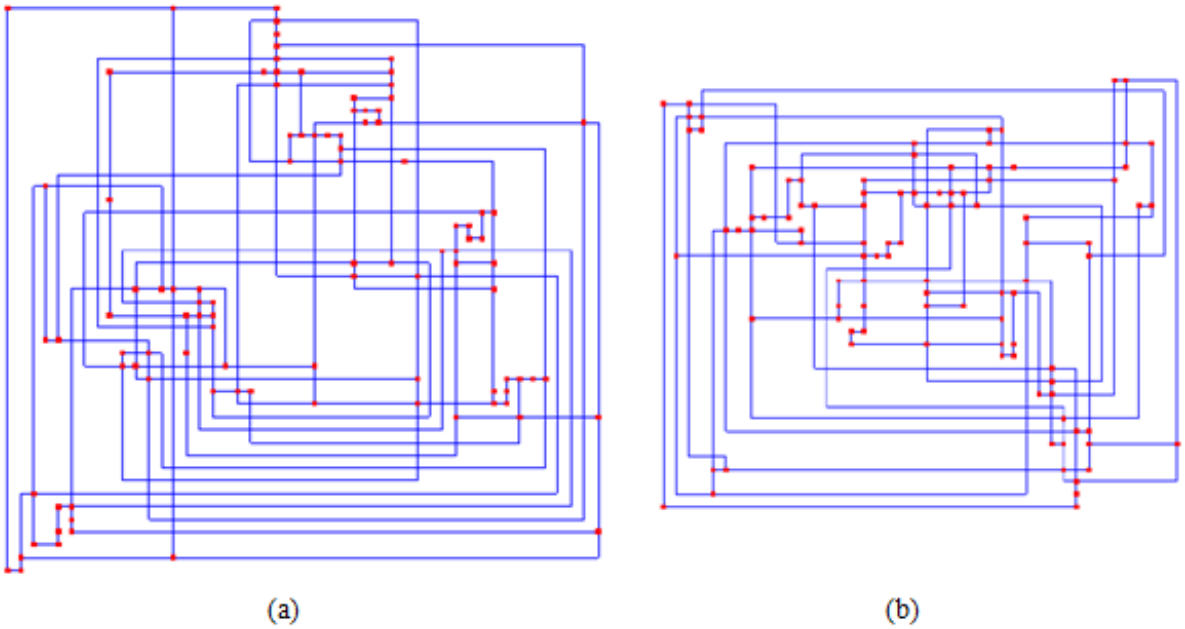


Figura 3.31 – Desenho final para o grafo com V = 100: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético utilizando a soma ponderada como função aptidão.

3.11.4. Resultados da abordagem híbrida TSM-FUZZY

Nesta seção, foram gerados os resultados considerando-se a abordagem híbrida, em que é utilizada a *tomada de decisão multicritério em ambiente fuzzy* no processo evolucionário, bem como na escolha do resultado final avaliado entre as 10 execuções do AG. A *Tabela 3.9* apresenta os resultados para o algoritmo genético para cada grafo de teste, identificado pelo número de vértices *V*. Na tabela, *Mu_D* é o grau de pertinência *fuzzy***.**

Tabela 3.9 - Resultados para o algoritmo genético usando a tomada de decisão *fuzzy* no processo evolucionário do AG.

Caso V-N	***Estatisticas***	f_X	f_B	f_L	***Mu_D***
10-30	**Melhor**	**0**	**3**	**22**	**1.000**
	Média	0	3	22	0.800
	Desvio padrão	0.00	0.00	0.52	0.26
20-30	**Melhor**	**1**	**1**	**45**	**1.000**
	Média	1	1	48	0.675
	Desvio padrão	0.00	0.00	0.95	0.24
30-30	**Melhor**	**1**	**5**	**83**	**1.000**
	Média	1	5	86	0.714
	Desvio padrão	0.00	0.00	1.69	0.25
40-30	**Melhor**	**5**	**5**	**160**	**0.857**
	Média	5	6	181	0.433
	Desvio padrão	0.52	0.00	4.37	0.26
50-30	**Melhor**	**12**	**10**	**225**	**0.769**
	Média	14	12	230	0.446
	Desvio padrão	1.69	1.96	7.66	0.23
100-30	**Melhor**	**62**	**31**	**320**	**0.917**
	Média	70	36	355	0.330
	Desvio padrão	7.89	3.30	8.29	0.31
150-30	**Melhor**	**86**	**36**	**393**	**1.000**
	Média	89	38	403	0.237
	Desvio padrão	5.56	3.08	25.33	0.29
180-30	**Melhor**	**95**	**51**	**531**	**0.696**
	Média	110	52	536	0.321
	Desvio padrão	10.27	2.17	35.14	0.26
200-30	**Melhor**	**334**	**120**	**687**	**0.698**
	Média	342	130	688	0.198
	Desvio padrão	18.57	5.91	34.21	0.26
250-30	**Melhor**	**427**	**145**	**910**	**0.848**
	Média	429	152	999	0.279
	Desvio padrão	12.77	4.47	77.81	0.35
500-30	**Melhor**	**319**	**103**	**1351**	**1.000**
	Média	321	108	1356	0.697
	Desvio padrão	2.67	6.98	6.86	0.41

As *Figuras 3.32 a 3.34* mostram os resultados, em termos de desenhos, para alguns dos casos testados na abordagem que utiliza a tomada de decisão *fuzzy* no processo

evolucionário. As figuras estão sempre acompanhadas dos desenhos correspondentes na abordagem clássica para fins comparativos.

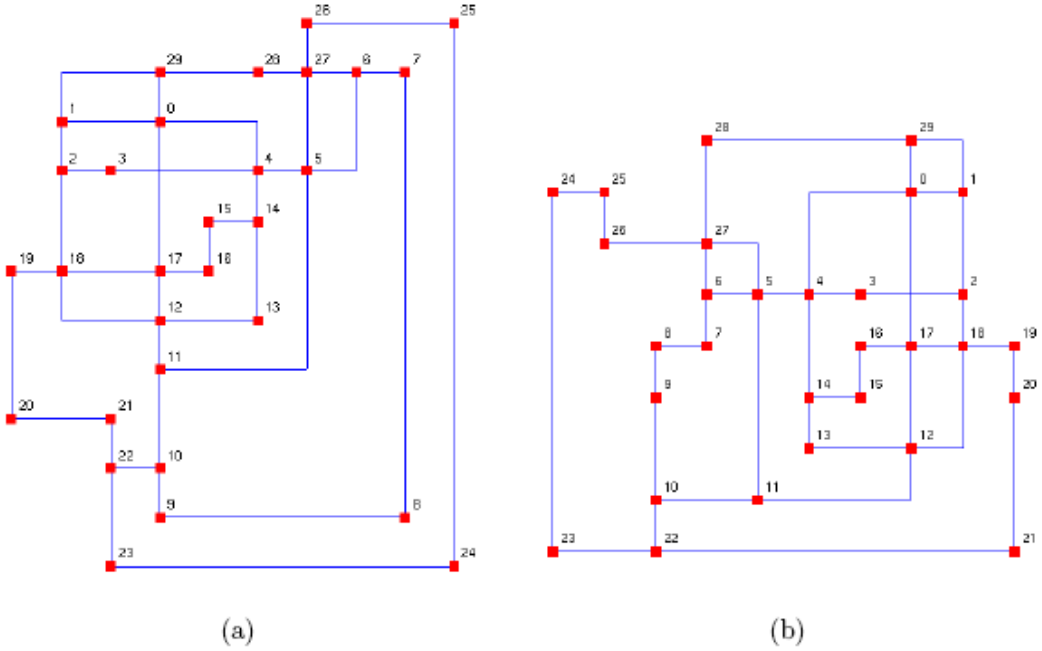


Figura 3.32 – Desenho final para o grafo com V = 30: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético utilizando tomada de decisão fuzzy.

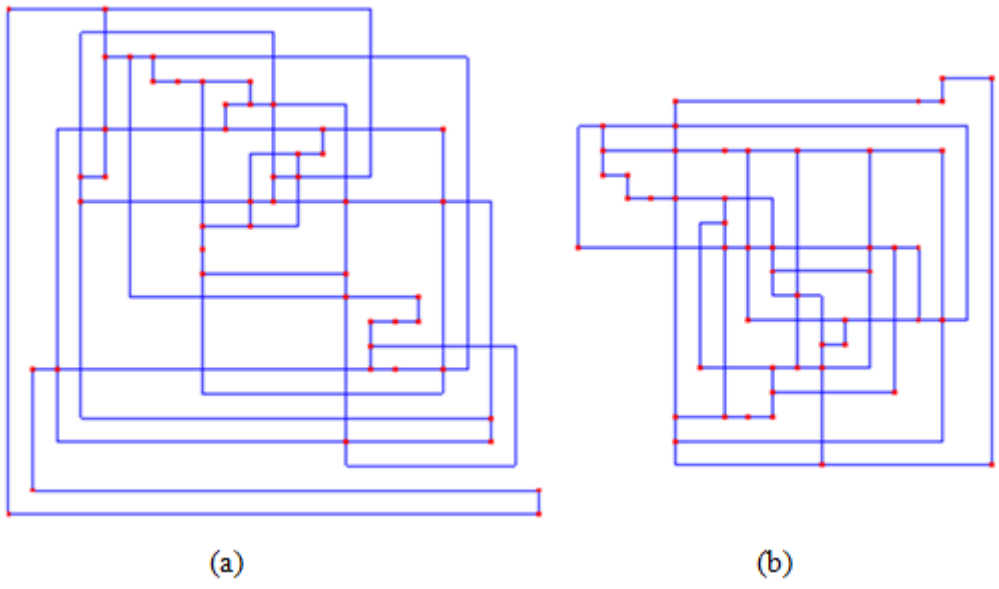


Figura 3.33 – Desenho final para o grafo com V = 50: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético utilizando tomada de decisão fuzzy.

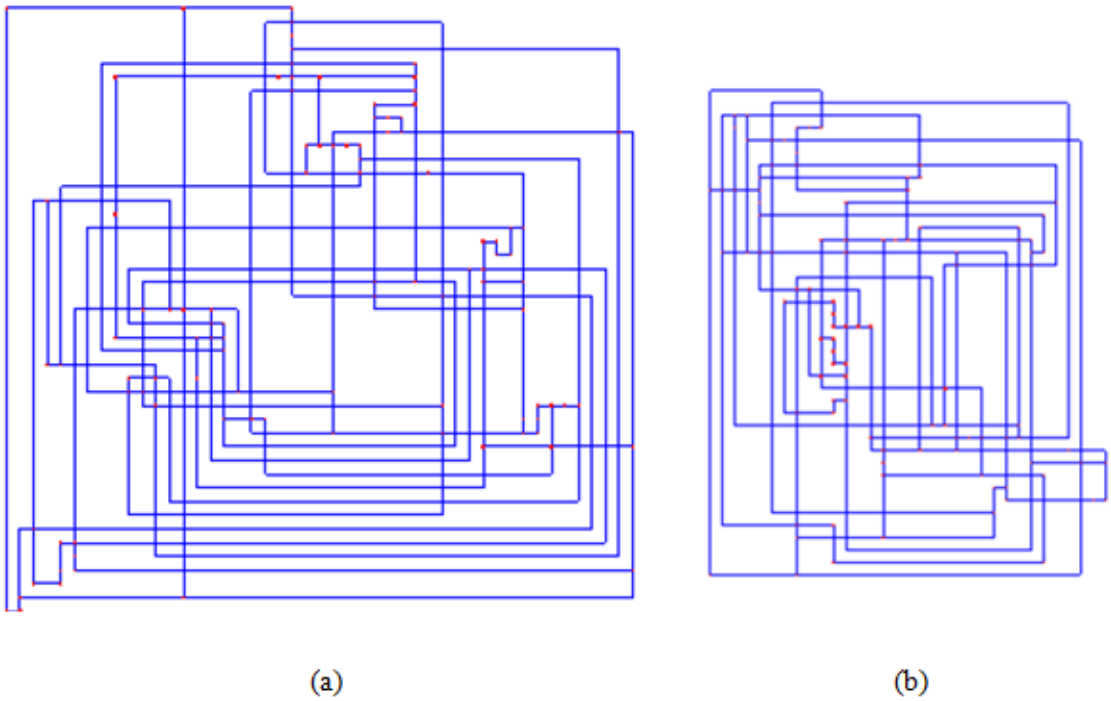


Figura 3.34 – Desenho final para o grafo com V = 100: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético utilizando tomada de decisão fuzzy.

Os resultados mostrados na *Tabela 3.9*, bem como os desenhos apresentados nas *Figuras 3.32 a 3.34*, mostram a superioridade da abordagem híbrida que considera a *topologia-forma-métrica* com o algoritmo genético, usando a *tomada de decisão multicritério em ambiente fuzzy* no processo evolucionário, em relação a abordagem clássica. Estes resultados foram escritos e submetidos ao periódico indexado internacional: Journal of Applied Soft Computing [126] em setembro de 2010.

3.11.5. Resultados da abordagem híbrida TSM-NSGAII

Nesta seção, são apresentados os resultados obtidos com a abordagem híbrida. Nela, é utilizado o algoritmo genético multiobjetivo NSGAII no processo evolucionário. Foi utilizada também a tomada de decisão *tomada de decisão multicriterio em ambiente fuzzy*, no final do processo para escolha de uma solução mais harmoniosa, dentre as diversas geradas na frente de Pareto F_1. Além disto, foi utilizada novamente a *tomada de decisão multicritério em ambiente fuzzy* para a escolha da solução mais harmoniosa dentre as 10 execuções do algoritmo genético NSGAII. A *Tabela 3.10* apresenta os resultados para algoritmo genético NSGAII para cada grafo de teste, identificado pelo número de vértices *V*.

Tabela 3.10 - Resultados para o algoritmo genético multiobjetivo NSGAII.

Caso V-N	***Estatisticas***	f_X	f_B	f_L
10-30	**Melhor**	**0**	**3**	**22**
	Média	0	3	23
	Desvio padrão	0.00	0.00	0.42
20-30	**Melhor**	**1**	**1**	**45**
	Média	1	2	47
	Desvio padrão	0.00	0.53	1.07
30-30	**Melhor**	**1**	**5**	**83**
	Média	1	5	87
	Desvio padrão	0.00	0.00	2.22
40-30	**Melhor**	**5**	**5**	**130**
	Média	5	5	145
	Desvio padrão	0.84	0.63	4.70
50-30	**Melhor**	**13**	**11**	**206**
	Média	14	11	213
	Desvio padrão	1.34	1.73	7.71
100-30	**Melhor**	**72**	**39**	**324**
	Média	73	38	307
	Desvio padrão	3.64	1.92	12.69
150-30	**Melhor**	**77**	**30**	**399**
	Média	87	39	411
	Desvio padrão	6.55	5.12	17.44
180-30	**Melhor**	**111**	**52**	**473**
	Média	116	54	497
	Desvio padrão	10.29	5.31	35.06
200-30	**Melhor**	**353**	**130**	**699**
	Média	357	132	700
	Desvio padrão	15.83	2.73	54.34
250-30	**Melhor**	**414**	**145**	**791**
	Média	388	143	1049
	Desvio padrão	44.19	9.58	134.55
500-30	**Melhor**	**317**	**102**	**1351**
	Média	321	110	1359
	Desvio padrão	3.71	7.86	7.06

As *Figuras 3.35 a 3.37* mostram os resultados, em termos de desenhos, para alguns dos casos testados na abordagem que utiliza o algoritmo genético multiobjetivo NSGAII. As

figuras estão sempre acompanhadas das figuras correspondentes na abordagem clássica para fins comparativos.

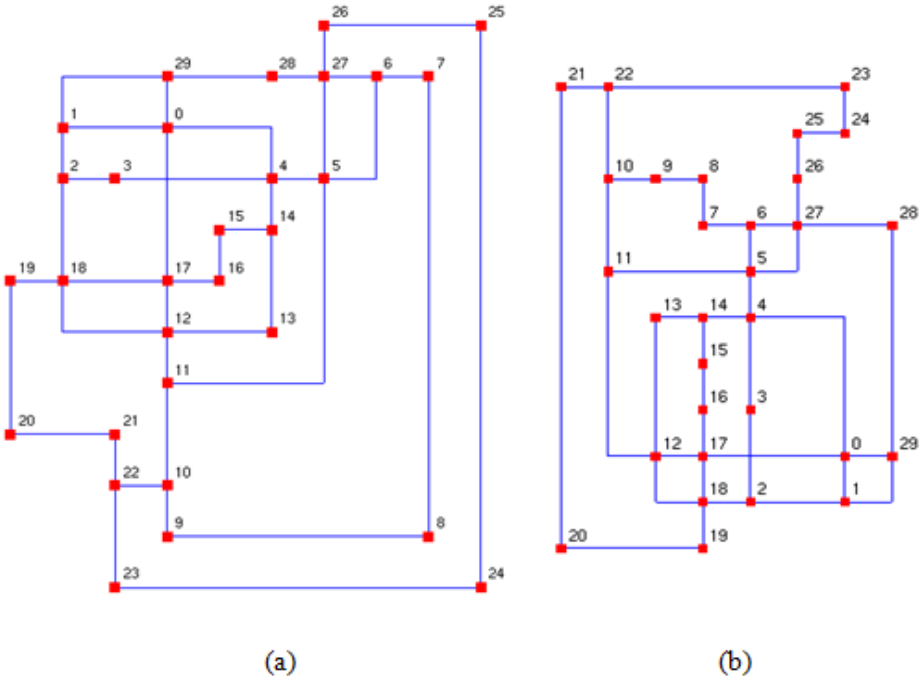


Figura 3.35 – Desenho final para o grafo com V = 30: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético multiobjetivo NSGAII.

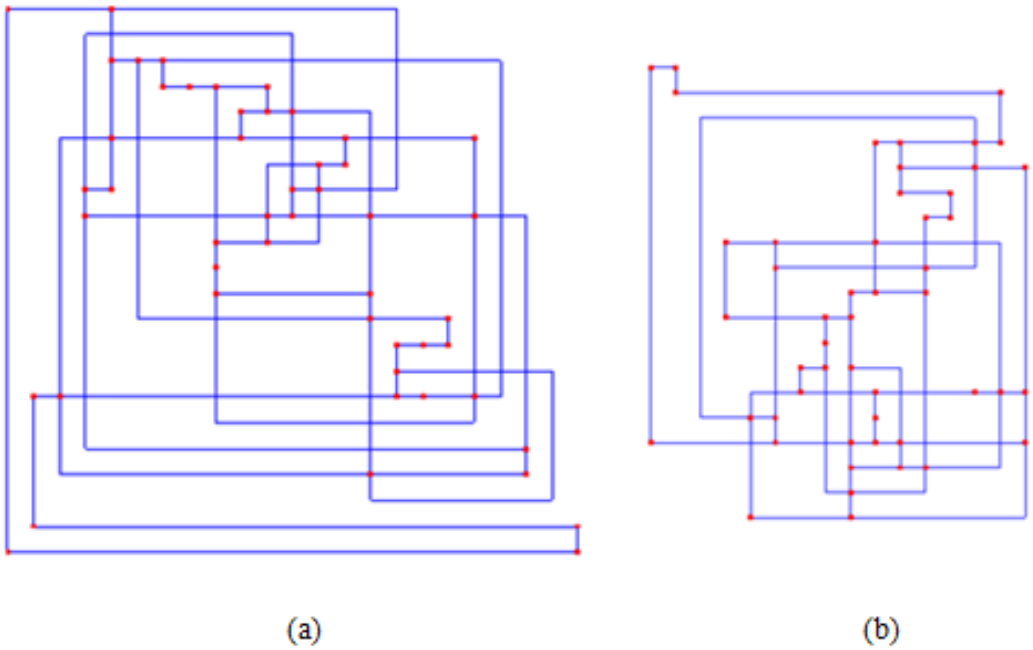
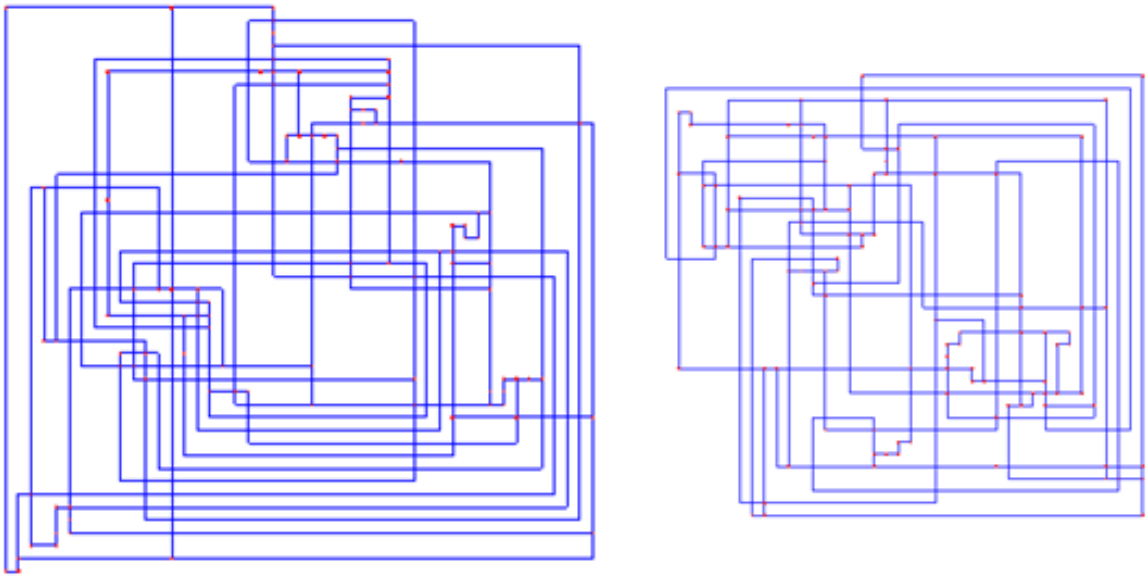


Figura 3.36 – Desenho final para o grafo com V = 50: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético multiobjetivo NSGAII.



(a) (b)

Figura 3.37 – Desenho final para o grafo com V = 100: (a) desenho obtido com a abordagem clássica, (b) desenho obtido com algoritmo genético multiobjetivo NSGAII.

Os resultados mostrados na *tabela 3.10*, bem como os desenhos apresentados nas *figuras 3.35 a 3.37*, mostram a superioridade da abordagem híbrida utilizando a *topologia-forma-métrica* com o algoritmo genético multiobjetivo NSGAII sobre a abordagem clássica.

Na próxima seção será feita uma análise dos resultados encontrados, tanto para a abordagem clássica como para as três abordagens híbridas desenvolvidas. Nesta seção os resultados são tabelados em uma única tabela (*Tabela 3.11*) para facilitar a análise conjunta de todos os resultados obtidos.

3.12. Análise dos resultados considerando as três abordagens híbridas

Na seção de resultados foi possível mostrar os ganhos significativos obtidos com as abordagens híbridas desenvolvidas. Os algoritmos genéticos, tanto aquele que utilizou a soma ponderada dos objetivos como função de aptidão, quanto o que utilizou a tomada de decisão multicritério em ambiente *fuzzy* no processo evolucionário, além daquele que utilizou o algoritmo genético multiobjetivo NSGAII, mostraram resultados superiores quando comparados aos obtidos pela abordagem tradicional. Estas abordagens híbridas proporcionaram resultados com ganhos de até 50% para o pior caso tratado. Desta maneira, foi possível encontrar resultados mais otimizados ao final do processo, garantindo, assim, um menor número de cruzamentos e um menor número de dobras, bem como um menor valor para a soma total dos tamanhos das arestas. Estas melhorias podem ser observadas

para os desenhos obtidos pelas três abordagens. O uso dos conceitos de tomada de decisão multicritério em ambiente *fuzzy*, tanto no processo evolucionário quanto no conjunto solução de Pareto (NSGAII), foi significativo para o sucesso das abordagens desenvolvidas, pois a obtenção de soluções, cujos níveis de satisfação dos objetivos possuem uma proximidade entre si, é garantida através do uso de tomada de decisão multicritério em ambiente *fuzzy*, o que nos assegura resultados considerados de qualidade do ponto de vista da harmonia, entre os níveis de satisfação de seus objetivos [113]. Além disto, a flexibilidade é agregada ao processo de forma a permitir o tratamento de diversos critérios estéticos (funções objetivo), bem como restrições, o que é importante. Deve-se ressaltar também a importância de poder utilizar qualquer tipo de função objetivo, independentemente do modelo em que representa, ou seja, se linear, não linear, etc. Isto é possibilitado pelo fato que o algoritmo genético não distingue as características do modelo utilizado.

A *Tabela 3.11* mostra os resultados para cada caso : clássico (TSM), aleatório e sem o uso do AG, abordagens híbridas TSM-WS, TSM-*FUZZY* e TSM-NSGAII para facilitar a análise conjunta dos resultados.

Tabela 3.11 - Resultados tabelados em conjunto para as abordagens clássica(TSM), aleatória sem o uso do AG, híbridas TSM-WS, TSM-*FUZZY* e TSM-NSGAII

-	-	***Clássico (TSM)***			***Aleatório sem o AG***			***TSM-WS***				***TSM-FUZZY***				***TSM-NSGAII***		
Caso V-N	***Estatisticas***	f_X	f_B	f_L	f_X	f_B	f_L	f_X	f_B	f_L	F	f_X	f_B	f_L	Mu_D	f_X	f_B	f_L
10-30	**Melhor**	**0**	**3**	**23**	**0**	**3**	**23**	**0**	**3**	**22**	**53**	**0**	**3**	**22**	**1.000**	**0**	**3**	**22**
	Média	0	3	23	0	3	24	0	3	23	55	0	3	22	0.800	0	3	23
	Desvio padrão	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.53	1.05	0.00	0.00	0.52	0.26	0.00	0.00	0.42
20-30	**Melhor**	**1**	**2**	**55**	**2**	**1**	**51**	**1**	**2**	**45**	**101**	**1**	**1**	**45**	**1.000**	**1**	**1**	**45**
	Média	1	2	55	1	2	55	1	1	47	103	1	1	48	0.675	1	2	47
	Desvio padrão	0.00	0.00	0.00	0.47	0.40	0.93	0.00	0.42	0.92	1.33	0.00	0.00	0.95	0.24	0.00	0.53	1.07
30-30	**Melhor**	**1**	**5**	**93**	**1**	**5**	**91**	**1**	**5**	**85**	**190**	**1**	**5**	**83**	**1.000**	**1**	**5**	**83**
	Média	1	5	93	1	5	93	1	5	87	196	1	5	86	0.714	1	5	87
	Desvio padrão	0.00	0.00	0.00	0.40	0.47	2.68	0.40	0.00	1.81	4.36	0.00	0.00	1.69	0.25	0.00	0.00	2.22
40-30	**Melhor**	**6**	**6**	**164**	**8**	**12**	**149**	**4**	**6**	**120**	**278**	**5**	**5**	**160**	**0.857**	**5**	**5**	**130**
	Média	6	6	164	8	11	154	6	6	123	291	5	6	181	0.433	5	5	145
	Desvio padrão	0.00	0.00	0.00	0.76	1.59	2.75	1.35	1.20	4.40	4.24	0.52	0.00	4.37	0.26	0.84	0.63	4.70
50-30	**Melhor**	**24**	**19**	**352**	**17**	**19**	**350**	**13**	**13**	**224**	**552**	**12**	**10**	**225**	**0.769**	**13**	**11**	**206**
	Média	24	19	352	19	17	359	14	12	234	573	14	12	230	0.446	14	11	213
	Desvio padrão	0.00	0.00	0.00	2.43	2.80	4.73	2.21	2.44	6.83	13.57	1.69	1.96	7.66	0.23	1.34	1.73	7.71
100-30	**Melhor**	**119**	**49**	**528**	**82**	**44**	**408**	**66**	**36**	**290**	**1018**	**62**	**31**	**320**	**0.917**	**72**	**39**	**324**
	Média	119	49	528	83	45	410	70	37	318	1102	70	36	355	0.330	73	38	307
	Desvio padrão	0.00	0.00	0.00	6.92	1.57	26.03	6.79	2.57	18.96	27.82	7.89	3.30	8.29	0.31	3.64	1.92	12.69
150-30	**Melhor**	**137**	**59**	**632**	**119**	**51**	**436**	**86**	**42**	**322**	**1200**	**86**	**36**	**393**	**1.000**	**77**	**30**	**399**
	Média	137	59	632	110	50	489	95	44	372	1351	89	38	403	0.237	87	39	411
	Desvio padrão	0.00	0.00	0.00	7.01	2.11	30.75	10.34	4.83	36.96	80.13	5.56	3.08	25.33	0.29	6.55	5.12	17.44

180-30	**Melhor**	**194**	**75**	**781**	**129**	**65**	**632**	**101**	**44**	**450**	**1537**	**95**	**51**	**531**	**0.696**	**111**	**52**	**473**
	Média	194	75	781	130	65	632	120	55	461	1689	110	52	536	0.321	116	54	497
	Desvio padrão	0.00	0.00	0.00	2.79	2.86	1.59	12.25	8.02	30.36	79.61	10.27	2.17	35.14	0.26	10.29	5.31	35.06
200-30	**Melhor**	**509**	**172**	**854**	**382**	**138**	**893**	**330**	**135**	**655**	**3365**	**334**	**120**	**687**	**0.698**	**353**	**130**	**699**
	Média	509	172	854	423	155	768	353	134	656	3477	342	130	688	0.198	357	132	700
	Desvio padrão	0.00	0.00	0.00	41.75	21.98	135.93	21.75	5.05	52.98	110.63	18.57	5.91	34.21	0.26	15.83	2.73	54.34
250-30	**Melhor**	**618**	**197**	**1144**	**495**	**159**	**1175**	**436**	**145**	**721**	**4057**	**427**	**145**	**910**	**0.848**	**414**	**145**	**791**
	Média	618	197	1144	497	170	1158	426	145	846	4256	429	152	999	0.279	388	143	1049
	Desvio padrão	0.00	0.00	0.00	2.87	9.92	13.68	31.70	7.75	121.28	184.69	12.77	4.47	77.81	0.35	44.19	9.58	134.55
500-30	**Melhor**	**608**	**173**	**2748**	**658**	**192**	**2148**	**317**	**102**	**1351**	**4593**	**319**	**103**	**1351**	**1.000**	**317**	**102**	**1351**
	Média	608	173	2748	609	201	2259	320	104	1371	4654	321	108	1356	0.697	321	110	1359
	Desvio padrão	0.00	0.00	0.00	84.59	15.83	190.65	5.31	2.42	24.50	81.79	2.67	6.98	6.86	0.41	3.71	7.86	7.06

A Tabela 3.11 facilita a análise comparativa dos resultados. Pode-se avaliar, por exemplo, que os resultados da abordagem clássica TSM são sempre inferiores aos da abordagem aleatória e das híbridas. A abordagem aleatória é intermediária em relação à clássica e às híbridas, o que mostra claramente a superioridade do algoritmo genético para a solução do problema. Em relação às abordagens híbridas, a soma ponderada dos objetivos, embora tenha mostrado bons resultados, apresenta dificuldades na escolha dos pesos ideais e não garante o equilíbrio entre os níveis de satisfação dos objetivos . A abordagem por tomada de decisão multicritério em ambiente fuzzy procura buscar este equilíbrio entre os níveis de satisfação dos objetivos garantindo assim soluções mais harmoniosas. E, finalmente, a abordagem puramente multiobjetivo nos garante a obtenção de um conjunto solução de Pareto, o que permite ao usuário a escolha de uma solução ideal de acordo com seus próprios critérios que são, na maioria das vezes, subjetivos. Com isto, podemos avaliar que as três metodologias trazem bons resultados e podem ser escolhidas de acordo com os interesses/necessidades de seus usuários. Além disto, seus resultados nos permitem validar as metodologias por mostrarem a obtenção de soluções iguais ou próximas entre si, obtidas através de técnicas diferentes.

De acordo com o descrito e os resultados obtidos, pode-se avaliar que os objetivos propostos para esta etapa do trabalho foram atingidos e novas metodologias baseadas em otimização foram desenvolvidas, representando contribuições importantes para a área de desenhos ortogonais de grafos no *grid*. Mais alguns detalhes importantes de se observar serão mostrados a seguir, bem como, também, desenhos que apresentam baixa qualidade.

As figuras 3.38, 3.39 e 3.40 mostram, para fins de análise comparativa, os desenhos para um mesmo grafo (grafo de 180 vertices), considerando a abordagem tradicional e as três abordagens desenvolvidas nesta etapa do trabalho. Nestas figuras, pode-se observar que, para todos os casos (quando comparados a abordagem clássica), os desenhos possuem melhor legibilidade e visibilidade, além de serem mais compactos. Especificamente para o caso em que foi utilizada a *tomada de decisão multicritério em ambiente fuzzy* (Figura 3.40) o desenho possui um aspecto visual mais suavisado, mais harmonioso.

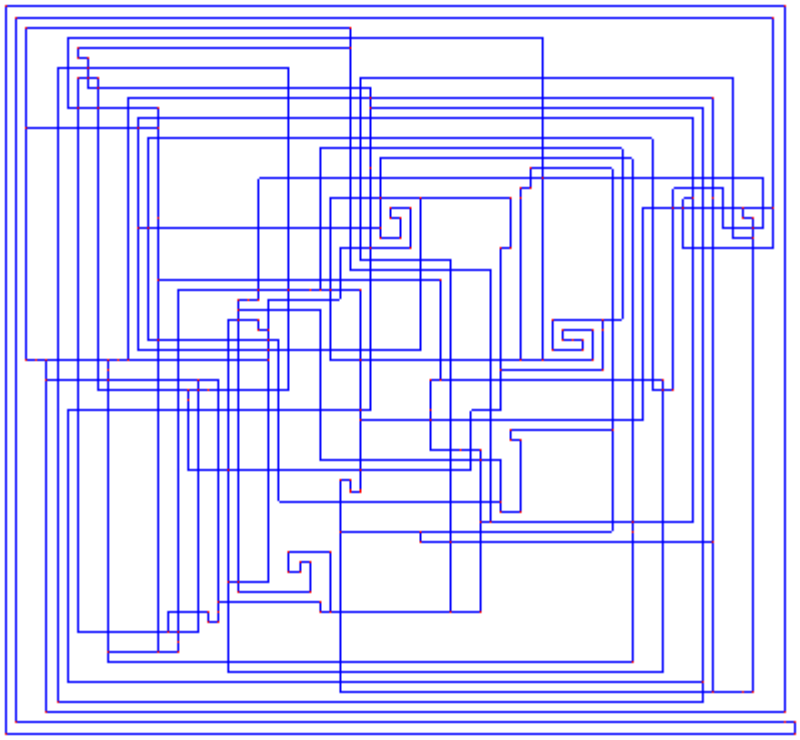


Figura 3.38 - Desenho final para o grafo com V = 180, utilizando a abordagem clássica.

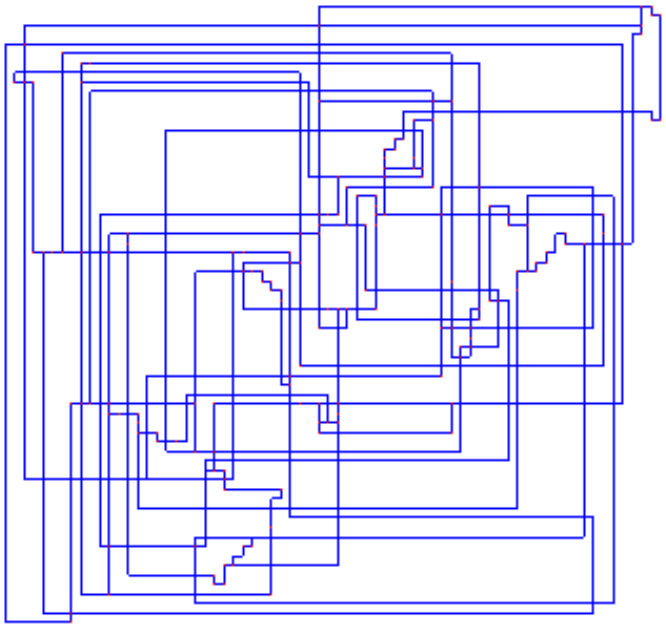
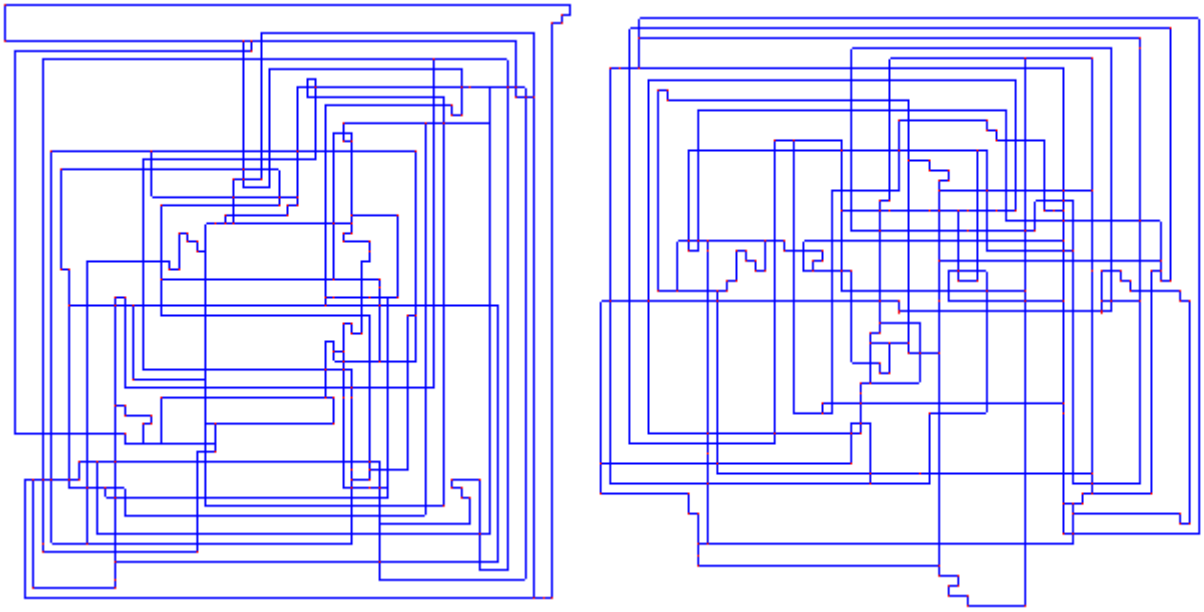


Figura 3.39 – Melhor desenho final para o grafo com V = 180, utilizando o algoritmo genético e a soma ponderada como função de aptidão.



(a) (b)

Figura 3.40 – (a) Melhor desenho final para o grafo com V = 180, utilizando o algoritmo genético e tomada de decisão fuzzy no processo evolucionário. (b) Melhor desenho final para o grafo com V = 180, utilizando o algoritmo genético multiobjetivo NSGAII.

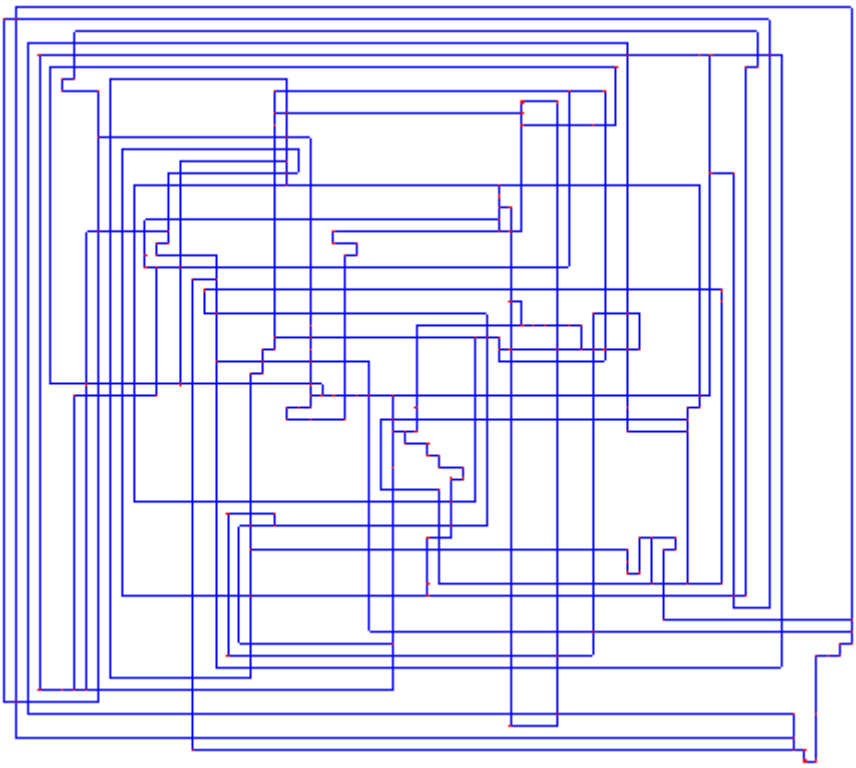


Figura 3.41 – (a) Desenho final para o grafo com V = 180, utilizando o algoritmo genético e a soma ponderada como função de aptidão para o pior caso obtido.

A Figura 3.41 mostra o desenho final para um grafo com 180 vértices, cujo resultado foi obtido utilizando o algoritmo genético e a soma ponderada dos objetivos. Neste caso os valores de f_X = 143, f_B = 60, f_L = 974 e F = 2843. Este é um resultado considerado de baixa qualidade do ponto de vista de todos os objetivos. Ele possui a soma total das arestas maior, consequentemente, possui uma área maior quando comparado o resultado da abordagem clássica (*Figura 3.38*).

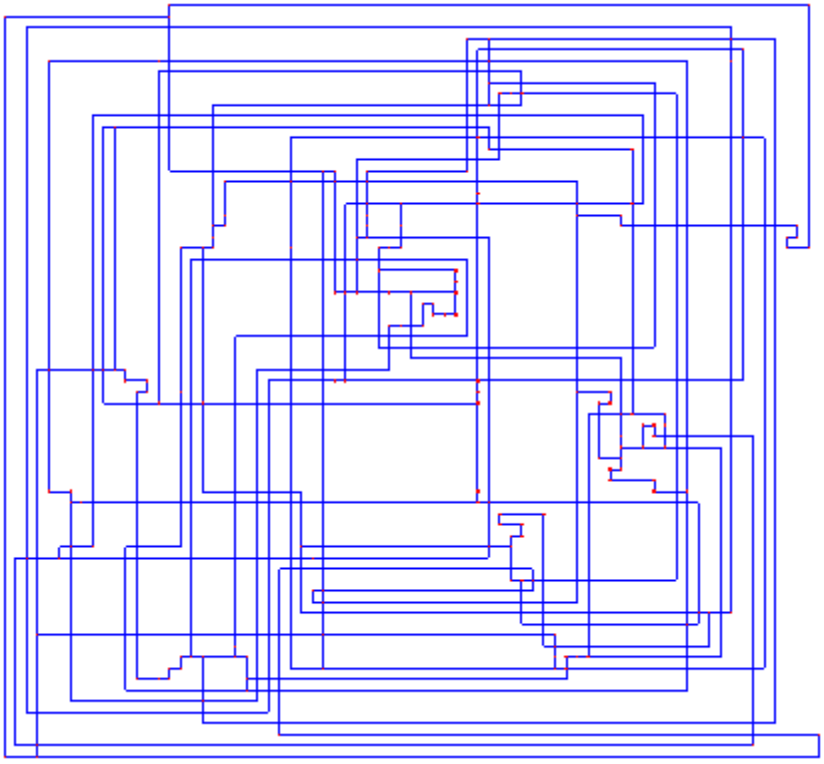


Figura 3.42 – (a) Desenho final para o grafo com V = 180, utilizando o algoritmo genético e a tomada de decisãomulticriterio em ambiente fuzzy no processo evolucionário para o pior caso obtido.

A Figura 3.42 mostra o desenho final para um grafo com 180 vértices, cujo resultado foi obtido utilizando o algoritmo genético e a tomada de decisão multicritério em ambiente *fuzzy* no processo evolucionário. Neste caso, os valores de f_X = 180, f_B = 74 e f_L = 816. Este é um resultado também considerado de baixa qualidade, do ponto de vista de todos os objetivos tratados pelo algoritmo genético.

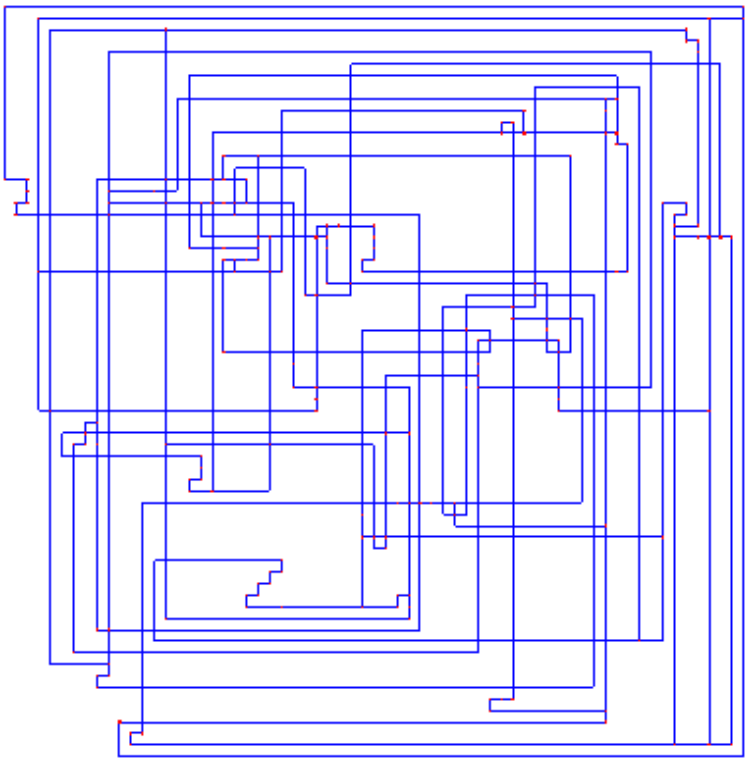


Figura 3.43 – (a) Desenho final para o grafo com V = 180, utilizando o algoritmo genético e a NSGAII para o pior caso obtido.

A *Figura 3.43* mostra o desenho final para um grafo com 180 vértices, cujo resultado foi obtido utilizando o algoritmo genético multiobjetivo NSGAII. Neste caso, os valores de f_X = 137, f_B = 56 e f_L = 565. Este é um dos resultados que compõem o conjunto solução de Pareto. Na ordenação das alternativas, utilizando-se tomada de decisão multicritério em ambiente *fuzzy*, este resultado está na quarta posição em um conjunto de 5 resultados. Portanto, este seria o quarto resultado mais harmonioso ou o segundo menos harmonioso em um conjunto de Pareto com cinco resultados obtidos pelo NSGAII.

Analisando os resultados através de seus desenhos, pode-se observar claramente que o algoritmo genético foi capaz de encontrar melhores deles para todas as três metodologias híbridas desenvolvidas, quando comparados aos obtidos pela abordagem clássica *topologia-forma-métrica*. Quando são selecionados os casos cuja solução possui mais baixa qualidade, entre as soluções obtidas pelo algoritmo genético, é que destaca-se a superioridade daquelas que são consideradas como melhores pelo AG. Quando ordenamos as alternativas pela metodologia de *tomada de decisão multicritério em ambiente fuzzy*, garantimos maior harmonia na satisfação dos objetivos, que caracteriza as soluções obtidas utilizando esta metodologia. Desta forma, foi possível validar a qualidade das metodologias desenvolvidas, uma vez que é possível compará-las com a abordagem

clássica, bem como entre elas próprias. Podemos observar uma melhoria que pode atingir até 50% em relação a abordagem clássica, quando comparadas ao pior caso tratado. O número de gerações nas abordagens híbridas variaram entre 5 a 25 no máximo. Um detalhe também importante de se ressaltar é que tratando o problema por três abordagens híbridas distintas, chegou-se a resultados muito próximos entre si. Isto permite considerar uma boa confiabilidade nas metodologias desenvolvidas, pois permite a sua validação com base nestas características.

Dando sequência aos desenvolvimentos, a fim de se atender aos objetivos deste trabalho, no próximo capítulo é descrita a nova abordagem desenvolvida para desenho automático de grafos ortogonais no *grid*, a abordagem unificada.

CAPÍTULO 4

ABORDAGEM UNIFICADA PARA DESENHO AUTOMÁTICO DE GRAFOS ORTOGONAIS

A abordagem clássica, *topologia-forma-métrica*, para desenhos de grafos ortogonais trata o desenho em três etapas, interdependentes, conforme mostrado anteriormente. Visando a eliminação das dependências entre essas etapas, uma abordagem unificada para desenho automático de grafos, que utiliza o algoritmo genético para resolver o problema de obtenção de desenhos ortogonais no *grid* em uma única etapa, é desenvolvida e apresentada neste capítulo. Os passos adotados para a obtenção dos desenhos são os seguintes:

- Descreve-se a estratégia para a obtenção do desenho ortogonal no *grid*;
- Definem-se as características do genoma que representa cada indivíduo (grafo) na população de indivíduos do algoritmo genético;
- Define-se a função de aptidão em função dos critérios estéticos a serem tratados e também, os pesos para cada critério estético, de acordo com a necessidade de penalização ou o grau de importância de cada um deles;
- Descrevem-se os operadores de seleção, cruzamento e mutação e os operadores de busca local utilizados no processo evolucionário do algoritmo genético para obtenção do desenho do grafo nesta nova abordagem.

4.1. Estratégia de desenho do grafo

Para se desenhar um grafo, a estratégia desenvolvida parte da representação do grafo através de sua lista de adjacências. Define-se um *grid*, com coordenadas inteiras, com tamanho suficiente para conter o desenho. O tamanho do *grid* deve ser função do número de vértices. *Grids* muito pequenos podem não ser suficientes para conter o desenho e muito grandes aumentam de maneira desnecessária o espaço de busca, elevando, por consequência, o tempo de processamento. Arbitrariamente, foi utilizado um *grid* com tamanho $2V$ x $2V$, onde V é o número de vértices do grafo. Este *grid* é representado por uma matriz $2V$ x $2V$, doravante denominada *matriz de checagem*. O tamanho ideal do *grid* pode ser definido, posteriormente, por experimentação.

A *Figura* 4.1 ilustra a construção da *matriz de checagem* para um grafo de 5 vértices em um *grid* de tamanho 10 x 10. Esta figura será utilizada para ilustrar a estratégia de desenho adotada nesta abordagem.

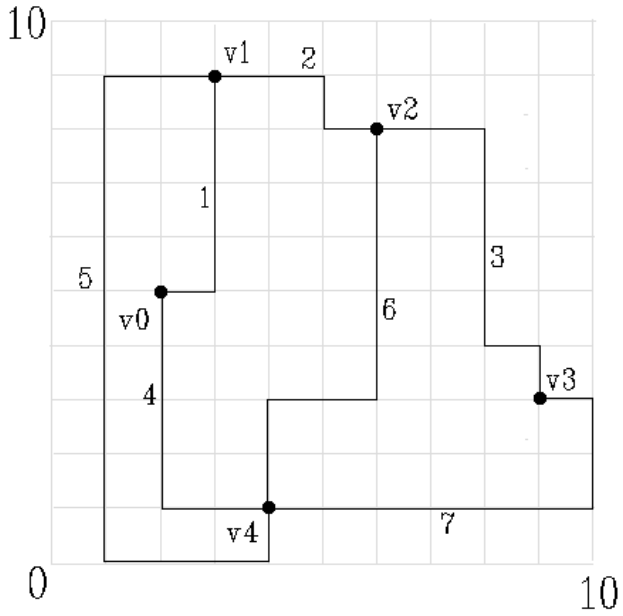


Figura 4.1 – Construção do desenho de um grafo de 5 vértices na matriz de checagem.

Nesta estratégia, o conjunto de vértices do grafo recebe os valores de suas coordenadas iniciais $C_v(x_v, y_v)$ aleatoriamente. Os valores das coordenadas variam entre os limites 0 a 2*V*. Estes vértices são, então, embutidos na *matriz de checagem* de acordo com as suas coordenadas. Veja, por exemplo, na *Figura* 4.1, os valores de coordenadas para os vértices $v_0(2,5)$ e $v_1(3,9)$.

Para cada aresta e_{ij} do grafo, é definido um vetor de bits com 3 bits. No primeiro bit é armazenada a informação a respeito da direção *d* de saída da aresta a partir do seu vértice de origem $i = C_v(x_i, y_i)$. Este bit vale 1 se a aresta sai na horizontal ou 0 se ela sai na vertical. Na *Figura* 4.1, veja a aresta que sai do vértice v_0 se conectando ao vértice v_1. O valor do primeiro bit (*d*) para esta aresta é 1, pois a mesma sai de v_0 na horizontal. Para definir se a aresta irá para direita/esquerda se a sua saída for pela horizontal ou para cima/baixo se a sua saída for pela vertical, utilizam-se as informações que estão armazenadas no segundo ou no terceiro bit do vetor de bits. Para mostrar como utilizar as informações relacionadas com o segundo e o terceiro bit, é necessário definir 4 variáveis denominadas Δ_i com $i = 1, ..., 4$, estas variáveis representam os tamanhos dos passos (o tanto em que se caminhará em *x* ou *y*) em relação à coordenada *x* ou *y*, ou seja, os tamanhos que serão atribuídos aos segmentos de aresta na construção do seu caminho em direção ao seu vértice de destino $j = C_v(x_j, y_j)$. Os valores de Δ_i são obtidos aleatoriamente para alguns casos e calculados para outros. No caso em que seja necessário calcular o valor de Δ_i, isto é feito considerando as equações:

$$\Delta x = x_j - x_i ; \quad (4.1)$$
$$\Delta y = y_j - y_i. \quad (4.2)$$

e obedecendo as restrições:

$$\Delta x = \sum_{i \in V} \Delta x_i ; \quad (4.3)$$
$$\Delta y = \sum_{i \in V} \Delta y_i . \quad (4.4)$$

Uma vez definido os Δ_i's , as suas equações e as suas restrições, será visto como são utilizados.

Considerando o segundo e o terceiro bit do vetor de bits, se o valor do segundo bit for igual a 1, indica que será considerado o valor de Δ_1 para efetuar o próximo passo e, se valer 0, significa que o mesmo não será utilizado. Se o valor do terceiro bit for igual a 1, indica que será considerado o valor de Δ_2 para efetuar o próximo passo e, se valer 0, significa que o mesmo não será utilizado. Pode-se também considerar os dois ao mesmo tempo, caso o vetor de bits contenha os valores 1 para o segundo e terceiro bit, ou mesmo nenhum dos dois, caso o vetor de bits contenha os valores 0 para o segundo e terceiro bits. Neste último caso, serão utilizados apenas Δ_3 e/ou Δ_4. A cada Δ_i utilizado haverá mudança na direção das arestas de forma alternada na horizontal e vertical.

A cada mudança de direção na construção do caminho da aresta, caracteriza-se uma dobra nesta aresta. Na estratégia adotada neste trabalho, fixa-se, arbitrariamente, o número total de dobras por aresta em no máximo 3 para não aumentar desnecessariamente o número de dobras do desenho. Além disto, um número de dobras maior que 3 pode levar a sobreposição de arestas na construção do caminho que conecta os vértices de origem e destino. A *Figura* 4.2 ilustra uma situação em que o número de dobras na aresta 1 é igual a 5 e a construção do caminho levou à sobreposição de 1 segmento da aresta, o que é indesejável em desenho de grafos.

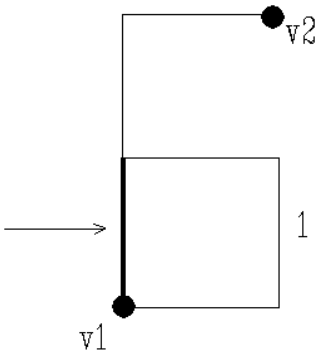


Figura 4.2 – Sobreposição de arestas devido ao número de dobras ser maior que 3.

Para arestas sem dobras, Δy ou Δx é nulo, ou seja, uma aresta sem dobras só poderá existir se uma das coordenadas dos vértices *i* e *j* forem idênticas (isto é, se $x_i = x_j$, ou se $y_i = y_j$), e os valores do segundo e terceiro bit no vetor de bits, que são relativos a Δ_1 e Δ_2 forem 0. Isto significa que Δ_1 e Δ_2 serão desconsiderados pelo vetor de bits. Então, para este caso, serão utilizados os valores calculados de Δ_3 ou Δ_4 para a construção do caminho, visto que é necessário apenas um passo para conectar dois vértices ligados por uma linha reta.

Para ilustrar os conceitos relacionados com a utilização do vetor de bits e dos valores dos passos (Δ_i's) vamos utilizar a *Figura* 4.1. Tomamos, por exemplo, a construção do caminho que conecta os vértices v_2 (6,8) a v_3(9,3). Este caminho é representado pela aresta *e* = 3. O vetor de bits para o primeiro segmento desta aresta é [1 1 1], os valores de Δ_1 = 2 e Δ_2 = -4. Como o bit responsável pela direção vale 1, a aresta sairá na horizontal, como o segundo bit vale 1, indica que será utilizado o tamanho de passo contido em Δ_1 = 2. Como este valor é positivo, a aresta seguirá para a direita em 2 unidades, assim a sua nova coordenada será *x = 8*. Como não se chegou ao vértice de destino ainda, esta nova coordenada *x* e a coordenada *y* anterior serão armazenadas como sendo as coordenadas da dobra C_d(8,8) que se formará ao mudar a direção no caminho da aresta, o que ficará definido com a construção do próximo segmento desta aresta. Respeitando a alternância na direção da aresta, a próxima será na vertical. Como o terceiro bit do vetor de bit vale 1, indica que será utilizado Δ_2 = -4, isto implica em dizer que se caminhará na vertical para baixo em 4 unidades. As coordenadas das dobras são geradas C_d(8,4). Se ainda não fechou o caminho os demais deltas (Δ_3 e Δ_4) são calculados de acordo com as equações (4.1 a 4.4.) e seus valores calculados são: Δ_3 = 1 e Δ_4 = -1. Os tamanhos destes passos são aplicados na construção do caminho, novas dobras são geradas e o caminho que conecta v_2 e v_3 é então completado.

A *Tabela4.1* mostra as possibilidades de combinações para os vetor de bits:

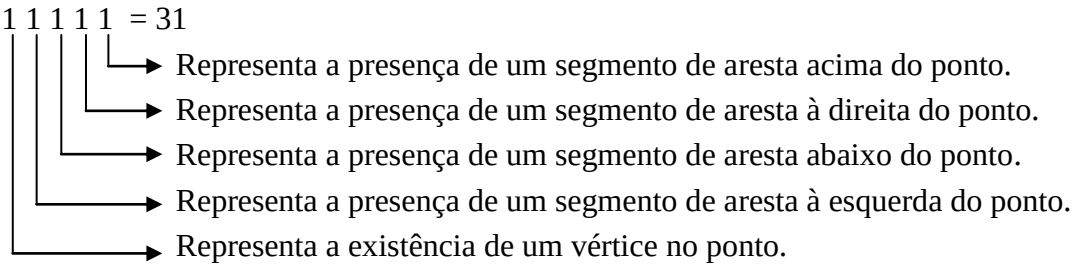
Tabela 4.1 – Possibilidade de combinações para o vetor de bits.

ítem	**d**	Δ_1	Δ_2	**Observação**
1	0	0	0	Aresta sai na vertical, não utiliza Δ_1 e nem Δ_2
2	0	0	1	Aresta sai na vertical, não utiliza Δ_1 e utiliza Δ_2
3	0	1	1	Aresta sai na vertical, utiliza Δ_1 e Δ_2
4	0	1	0	Aresta sai na vertical, utiliza Δ_1 e não utiliza Δ_2
5	1	0	0	Aresta sai na horizontal, não utiliza Δ_1 e nem Δ_2
6	1	0	1	Aresta sai na horizontal, não utiliza Δ_1 e utiliza Δ_2
7	1	1	1	Aresta sai na horizontal, utiliza Δ_1 e Δ_2
8	1	1	0	Aresta sai na horizontal, utiliza Δ_1 e não utiliza Δ_2

É importante enfatizar que os valores de Δ_1 e Δ_2 são sempre obtidos aleatoriamente e os valores de Δ_3 e Δ_4 são sempre calculados de acordo com as *Equações* 4.1 e 4.2 respeitando as restrições das *Equações* 4.3 e 4.4.

Durante o processo de contrução do desenho, todas as informações sobre a construção são armazenadas de maneira a permitir o controle das mesmas quando inseridas na *matriz de checagem*. Assim, cada valor na matriz de checagem representa as coordenadas *x* e *y* de um ponto que pode conter um vértice, um segmento de aresta ou ambos, com as informações sobre a sua vizinhança. Estas informações são representadas por um valor inteiro que utiliza a representação binária de 5 bits, onde cada bit carrega consigo uma informação sobre a ocupação da quadrícula atual e das quadrículas da matriz de checagem em torno daquele ponto. Desta maneira, uma quadrícula da matriz é representada por:

Matriz[i][j] =



Com esta representação é possível controlar todas as informações contidas no *grid* e, com isto, calcular o número de cruzamentos entre arestas, número de dobras existentes na aresta, sobreposições (de arestas, de arestas-vertices e de vértices) e a soma total das arestas. Estes valores irão compor a função de aptidão utilizada na estratégia evolucionária do algoritmo genético.

É importante definir como as informações sobre o grafo, que no contexto do algoritmo genético é intitulado genoma ou indivíduo, serão representadas.

4.1.1. Representação do genoma

O genoma será composto por informações referentes aos vértices e arestas do grafo, como segue:

$$(C_v\ (x_v, y_v)\ ,\ e\ ([0\ 0\ 1],\ \Delta_1,\ \Delta_2)) \tag{4.1}$$

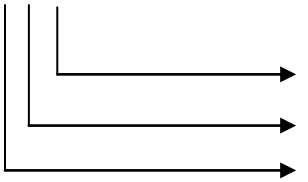
onde:

$C_v\ (x_v, y_v)$ = coordenadas dos vértices do grafo *G*;

$e_{ij}\ ([0\ 0\ 1],\ \Delta_1,\ \Delta_2)$ = para cada aresta vincular as informações;

Δ_1 e Δ_2 representam o tamanho do passo que será dado na horizontal ou na vertical, quando da formação do caminho percorrido pelas arestas na ligação que fará com os seus vértices de origem e destino.

[0 0 1] = vetor de bits que representam as seguintes informações:



Se valor = 1 então utilizar Δ_2, se valor = 0 então não utilizar Δ_2.

Se valor = 1 então utilizar Δ_1, se valor = 0 então não utilizar Δ_1.

Se valor = 1 então aresta sairá pela horizontal, se valor = 0 então aresta sairá pela vertical;

Na próxima é será definida a função de aptidão utilizada pelo algoritmo genético.

4.1.2. Função de aptidão

+.

As funções objetivo (critérios estéticos) a serem considerados nesta abordagem de desenho ortogonal de grafos são listadas a seguir:

f_{SE} **=** número de sobreposições de arestas**;**
f_{SEV} **=** número de sobreposições de arestas e vértices**;**
f_{SV} **=** número de sobreposições de vértices**;**
f_X **=** número de cruzamentos entre arestas**;**
f_B **=** número de dobras**;**
f_L **=** soma total dos tamanhos das arestas;

Nesta abordagem, por escolha, será primeiramente adotada a soma ponderada dos objetivos de maneira a trasformar o problema multiobjetivo em um problema mono-objetivo. Esta soma ponderada será utilizada como função de aptidão no processo evolucionário do algoritmo genético e é mostrada a seguir:

$$min\ \boldsymbol{F} = \alpha_1 \bullet (f_{SE} + f_{SEV}) + \alpha_2 \bullet f_{SV} + \alpha_3 \bullet f_X + \alpha_4 \bullet f_B + \alpha_5 \bullet f_L \qquad (4.5)$$

Os valores de α_i são os pesos ou penalidades definidos experimentalmente para cada critério estético considerado. Seus valores são:

α_1 = 3000; α_2 =12000; α_3 = 800; α_4 = 500 e α_5 = 300

Deve-se dar maior importância para se evitar sobreposições de arestas, de arestas-vertices e de vértices na estratégia de desenho adotada nesta abordagem, uma vez que as mesmas são indesejáveis no desenho final. Por esta razão, é aplicada uma penalidade

grande o suficiente para evitar tais sobreposições (α_1 e α_2). Estes valores foram definidos experimentalmente. Respeitando a teoria relacionada com a qualidade de desenho de grafos [18], [127] que priorizam os critérios estéticos de forma a prover melhor visibilidade e legibilidade ao desenho, foram escolhidos também experimentalmente, os valores dos pesos para α_3, α_4 e α_5 de forma que $\alpha_3 > \alpha_4 > \alpha_5$.

4.3. Operadores para o algoritmo genético

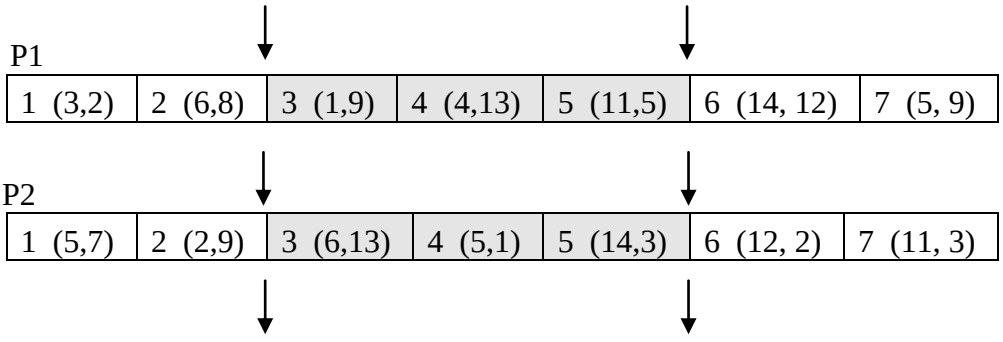
Os operadores de seleção, cruzamento e mutação que compõem o algoritmo genético na abordagem unificada são aqui, descritos. Os operadores desenvolvidos para esta abordagem foram adaptados a partir de conceitos estudados na literatura [34] para a solução do problema de desenho ortogonal de grafos. Os mesmos serão descritos a seguir:

a) **Operador de seleção:** a implementação do algoritmo genético empregou seleção por torneio binário.

b) **Operador de cruzamento:** para efetuar os cruzamentos no genoma é necessário considerar cada parte do mesmo em separado. Para as coordenadas dos vértices, o cruzamento é realizado utilizando-se duas técnicas denominadas *cruzamento discreto de vértices* e *cruzamento de vértices por médias*. Para o vetor de bits e para os Δ`s é utilizado o *cruzamento de arestas*. Essas técnicas são aplicadas em uma proporção pré-definida de indivíduos a sofrerem cruzamentos. Esses operadores são descritos a seguir: ‘s

   - **Para as coordenadas dos vértices:**

     - **Cruzamento discreto de vértices:** seleciona-se, aleatoriamente, dois pontos de corte no mapa de coordenadas de vértices. Os vértices do primeiro filho pertencentes a este intervalo recebem coordenadas do pai_1 e, fora dele, recebem coordenadas do pai_2. Depois o processo se inverte: os vértices neste mesmo intervalo, para o segundo filho, recebem as coordenadas do pai_2 e, fora dele, as coordenadas do pai_1.



F1

1 (5,7)	2 (2,9)	3 (1,9)	4 (4,13)	5 (11,5)	6 (12, 2)	7 (11, 3)

F2

1 (3,2)	2 (6,8)	3 (6,13)	4 (5,1)	5 (14,3)	6 (14, 12)	7 (5, 9)

- **Cruzamento de vértices por médias:** seja uma variável $\lambda \in (0,1)$, para as informações dos vértices com coordenadas (x, y), os novos valores de coordenadas dos vértices são obtidos de acordo com as seguintes equações:

$$x' = \lambda x_i + (1-\lambda)x_j \quad (4.6)$$

$$y' = \lambda y_i + (1-\lambda)y_j \quad (4.7)$$

onde:

$\lambda = \psi \bullet (2\varepsilon + 1) - \varepsilon$;

$\psi \in (0,1)$ → obtido aleatoriamente por uma função de densidade de probabilidade uniforme (rand());

$\varepsilon > 0$ → o tamanho que se deseja aumentar na zona de busca do cruzamento (veja figura 4.1);

x' = coordenada x do descendente;

y' = coordenada y do descendente;

x_i = coordenada x do primeiro pai;

x_j = coordenada x do segundo pai;

y_i = coordenada y do primeiro pai;

y_j = coordenada y do segundo pai;

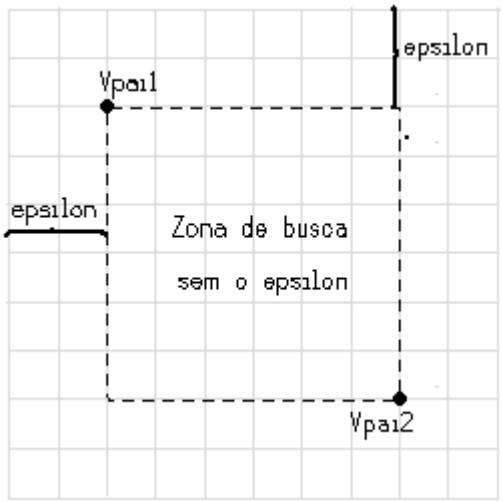


Figura 4.3 – Zona de busca em relação ao vértice do pai$_1$ e o vértice do pai$_2$ antes e depois do acréscimo do epsilon.

Desta forma, podemos definir as equações para o cruzamento das coordenadas dos

vértices como sendo:

$$x_{offspring1} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai1} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai2} \quad (4.8)$$

$$x_{offspring2} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai2} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai1} \quad (4.9)$$

$$y_{offspring1} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai1} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai2} \quad (4.10)$$

$$y_{offspring2} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai2} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai1} \quad (4.11)$$

Com $\varepsilon \rightarrow 20\% = 0.2$.

Os valores para as novas coordenadas devem ser arredondados para o inteiro mais próximo. De posse das novas coordenadas dos vértices a *matriz de checagem* é reconstruída.

- **Cruzamento de arestas**

O cruzamento das arestas é feito considerando o vetor de bits das arestas bem como os valores de Δ_1 e Δ_2 conforme mostrado a seguir:

- **Para os vetores de bits:**

Seleciona-se um único ponto de corte no vetor de bits, aleatoriamente. O primeiro filho recebe os bits do pai_1 até o ponto de corte, e recebe os do pai_2, depois do ponto de corte. O segundo filho recebe os bits do pai_2 até o ponto de corte, e recebe os do pai_1, depois do ponto de corte.

P1 ↓	P2 ↓		F1	F2
[0 1 0]	[1 0 1]	→	[0 1 1]	[1 0 0]

- **Para os valores de Δ_1 e Δ_2:**

O mesmo raciocínio desenvolvido para o cruzamento de vértices por médias é aplicado para o cruzamento dos Δ's.

$$\Delta_{1\,offspring1} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{1\,pai1} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{1\,pai2} \quad (4.12)$$

$$\Delta_{1\,offspring2} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{1\,pai2} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{1\,pai1} \quad (4.13)$$

$$\Delta_{2\,offspring1} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{2\,pai1} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{2\,pai2} \quad (4.14)$$

$$\Delta_{2\,offspring2} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{2\,pai2} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet \Delta_{2\,pai1} \quad (4.15)$$

` Com $\varepsilon \rightarrow 20\% = 0.2$.

Os valores para os novos Δ's devem ser arredondados para o inteiro mais próximo.

c) **Operador de mutação:** o operador de mutação utilizado é baseado na distribuição normal (Gaussiana), em que são requeridos dois parâmetros: a média μ e o desvio padrão σ. A mutação é realizada através da adição de um valor de "perturbação" nos elementos a serem mutados, que, no caso em estudo, são as coordenadas dos vértices, o vetor de bits e os Δ_1 e Δ_2 das arestas. Os valores da "perturbação" são aleatoriamente obtidos usando a distribuição Gaussiana $N(\mu,\sigma)$ [37]. Na prática, a média μ é sempre definida com o valor 0. A seguir serão definidos os operadores de mutação de vértices e de arestas com base neste conceito.

- **Operador de mutação de vértices:**

  As coordenadas dos novos vértices serão obtidas com base na distribuição Gaussiana conforme mostrado a seguir:

$$x' = x + N(\mu,\sigma) \tag{4.16}$$
$$y' = y + N(\mu,\sigma) \tag{4.17}$$

onde:

x' = coordenada *x* após a mutação
x = coordenada *x* antes da mutação
y' = coordenada *y* após a mutação
y = coordenada *y* antes da mutação

$N(\mu,\sigma)$ = distribuição de probabilidade normal (Gaussiana) com média $\mu = 0$ e desvio padrão σ = 10% ou 20% (0.1 ou 0.2) do tamanho do *grid* que vale 2*V*, ou seja, σ = 0.1 • 2*V* ou 0.2 • 2*V*.

- **Operador de mutação de arestas:**

  Para cada aresta *e* que sofrerá a mutação, é atribuída uma "perturbação" com base na distribuição Gaussiana citada, sobre os valores de Δ_1 e Δ_2, além disto, inverte-se um dos bits do vetor de bits, escolhido aleatoriamente.

$$\Delta_1' = \Delta_1 + N(\mu,\sigma) \tag{4.18}$$
$$\Delta_2' = \Delta_2 + N(\mu,\sigma) \tag{4.19}$$

  É feita a inversão de um bit no vetor de bits da aresta correspondente. Por exemplo: [0 1 0] → [1 1 0].

d) **Elitismo**

O elitismo é uma técnica utilizada para melhorar a convergência dos algoritmos. A técnica consiste em se escolher o melhor (ou melhores) indivíduo(s) de cada geração e garantir uma cópia dele(s) na geração seguinte sem que o mesmo (ou mesmos) sofra(m) cruzamentos e mutações, de forma a conservar suas características consideradas boas. A utilização do elitismo faz com que o algoritmo convirja mais cedo, porque evita que os elementos mais aptos sejam perdidos ou modificados durante as gerações [37]. É utilizada a técnica de elitismo na abordagem unificada para desenhos ortogonais de grafos.

e) **Busca Local**

Foram desenvolvidos também operadores de busca local que compõem o algoritmo genético e auxiliam na busca por novas soluções, dificilmente encontradas, caso se utilizasse somente os operadores globais. São eles:

a) Polarização de arestas e deltas;
b) Compactação;
c) Melhoria local;

A polarização de arestas consiste em ordenar as arestas do indivíduo em ordem decrescente em relação ao número de sobreposições, tomar os primeiros 20% das arestas ordenadas e aplicar nelas o procedimento de polarização descrito a seguir:

Testar se a aresta está saindo junto de outra aresta no seu vértice de origem;
Caso positivo: verificar na matriz onde, em torno do vértice, existe um espaço vazio e fazer com que a aresta saia por este espaço vazio;
Caso negativo: mudar as características da aresta de forma a evitar a subreposição:

i. 33% de chance mudando qualquer bit de seu vetor de bits;
ii. 33% de chance mudando o valor de Δ_1;
iii. 33% de chance mudando o valor de Δ_2;

A polarização dos deltas consiste em tomar cada aresta a ser polarizada (20% das que possuem maior número de sobreposições) e aplicar nelas variações nos valores dos deltas, checando quais variações geram menor valor de aptidão para aquele indivíduo.

A compactação consiste em ordenar os valores das coordenadas *x* e *y* do grafo *G* e os vértices que compartilham as coordenadas *x* na extrema direita têm suas coordenadas *x* decrescidas de uma unidade, em um loop, até que esse decréscimo não melhore mais o desenho. Os vértices que compartilham as coordenadas y na extrema superior têm suas

coordenadas *y* decrescidas de uma unidade, em um loop, até que esse decréscimo não melhore mais o desenho. Os vértices que compartilham as coordenadas *x* na extrema esquerda têm suas coordenadas *x* acrescidas de uma unidade em um loop até que esse acréscimo não melhore mais o desenho. Os vértices que compartilham as coordenadas *y* na extrema inferior têm suas coordenadas *y* acrescidas de uma unidade, em um loop até que esse acréscimo não melhore mais o desenho. Este procedimento é feito de modo a compactar o desenho da direita para a esquerda, da esquerda para direita, de cima para baixo e de baixo para cima.

A melhoria local consiste em tomar o melhor indivíduo e avaliar se é possível conseguir alguma melhoria a mais nele. Isto é feito tomando cada vértice do grafo e provocando uma perturbação de uma unidade nas coordenadas dos seus vértices, em todas as direções, mantendo sempre aquela que gerar melhores valores de aptidão para o grafo.

Na próxima seção será feita uma síntese da abordagem unificada desenvolvida e seu algoritmo principal será descrito. Os algoritmos e operadores de busca local são detalhados no anexo I.

4.4. Síntese do funcionamento da abordagem unificada e seus algoritmos

A bordagem unificada é sintetizada como segue: define-se, inicialmente, o tamanho da população inicial (*N)*. Os *N* indivíduos que compõem a população inicial são gerados. O genoma é construído pelas coordenadas dos vértices e pelos valores que compõem o vetor de bit (direção e os valores dos deltas). As características de cada indivíduo da população são inseridas na matriz de checagem, que representa o *grid*, onde o grafo toma efetivamente a sua forma. Neste momento, as informações necessárias para compor a função de aptidão (*Equação 4.5*) são obtidas. Uma vez calculados os valores que compõem a função de aptidão, a população está pronta para iniciar o processo evolucionário. São aplicados os operadores do algoritmo genético (seleção, cruzamento e mutação) descritos anteriormente. Cada indivíduo é submetido a matriz de checagem novamente para que os novos indivíduos tomem sua forma e sua função de aptidão seja calculada novamente. Neste momento, 20% da nova população é submetida aos operadores de busca local (polarização das arestas e dos deltas, compactação e melhoria local) e o melhor indivíduo é selecionado. Este melhor indivíduo terá sua presença garantida na próxima população. Assim, o processo se inicia novamente até que o critério de parada seja atingido.

O critério de parada considerado é um número de gerações grande o suficiente para se garantir a convergência do algoritmo genético. Para se chegar ao valor deste número, foram armazenados, experimentalmente, os valores médios de aptidão calculados durante o

processo evolucionário, e com eles, um gráfico de convergência foi construído. Neste gráfico foi possível detectar o momento em que não mais ocorria melhorias nos valores médios de aptidão. Com isto, foi possível definir o número de gerações necessárias para se garantir a convergência do algoritmo genético para cada caso dos grafos de teste. Um outro critério possível é o utilizado na abordagem híbrida. Neste caso, seria observada a não melhoria no melhor indivíduo por um número *M* de gerações, com *M* grande o suficiente para se assegurar que não haveria espaço para mais melhorias.

O algoritmo é mostrado a seguir. Os algoritmos chamados por ele são detalhados no anexo I.

AlgoritmoUnificado (AG-DG-UNIFIED)

Entrada: grafo *G;*

Saida: desenho ortogonal otimizado;

1 - Geração da População inicial:

N = tamanho da população de individuos*;*

a) Gere a população inicial: os *N* indivíduos tem genoma aleatório.

2 – Processo evolucionário do AG

Enquanto (não atingiu o critério de parada **) faça**

{

- Aplique os operadores do AG (seleção, cruzamento e mutação) na população inicial;
- Para cada indivíduo da população inicial, construa a *matriz de checagem* e avalie a sua função de aptidão (*Equação* 4.5);
- Aplique os operadores de busca local em 20% da população (polarização de arestas e deltas, compactação e melhoria local);
- Calcule novamente os valores da função de aptidão de cada novo indivíduo;
- Armazene o melhor indivíduo encontrado até então (elitismo);

}

3 – Armazene o melhor indivíduo encontrado de acordo com os valores da função de aptidão.

4.5. Resultados da abordagem unificada

Nesta seção, são apresentados os resultados obtidos através da aplicação das técnicas desenvolvidas para o desenho ortogonal de grafos na abordagem unificada. Estes resultados consideram a utilização da *Equação* (*4.5*) como a soma ponderada dos critérios estéticos tratados nesta abordagem.

A *Tabela 4.2* mostra os resultados obtidos para cada objetivo considerado neste trabalho, bem como o valor da função de aptidão com base na *Equação* (*4.5*). Na tabela, f_{SE} representa as sobreposições entre arestas, f_X representa o número de cruzamentos do desenho, f_B representa o número de dobras, f_L a soma total das arestas, F a soma ponderada dos critérios estéticos, NG o número de gerações do AG para o caso de teste e $T(s)/NG$ a relação entre o tempo total gasto (em segundos) e o número de gerações do AG para encontrar o melhor resultado em determinada execução do AG.

Tabela 4.2 - Resultados para a abordagem Unificada.

V	f_{SE}	f_X	f_B	f_L	F	NG	$T(s)/NG$
10	0	0	3	22	8100	100	0.1699
20	0	1	1	37	12400	300	2.0667
30	0	5	9	77	31600	2500	48.5664
40	0	10	10	159	60700	2250	76.8053
50	0	29	22	337	135300	2000	93.3637
60	0	79	38	619	267900	2000	130.8255
80	5	201	63	1288	593700	4500	173.0440
100	11	313	76	2245	1003900	1500	245.7743

O gráfico de tempo computacional para o algoritmo genético da abordagem unificada, por geração, em relação ao número de vértices, utilizando a soma ponderada dos objetivos como função de aptidão é mostrado na *Figura 4.4*.

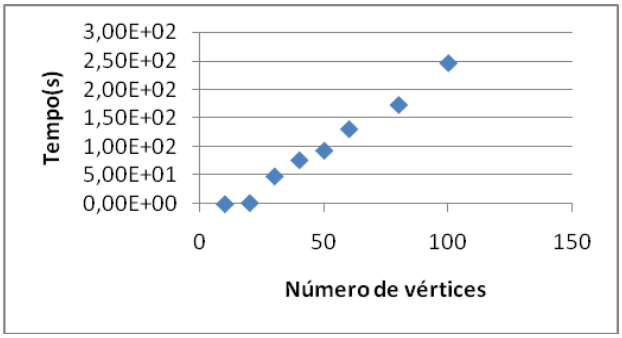


Figura 4.4 – Tempo computacional para o algoritmo genético da abordagem unificada por geração em relação ao número de vértices dos grafos utilizando a soma ponderada dos objetivos como função de aptidão.

Os desenhos obtidos são mostrados a seguir:

A *Figura 4.5* mostra o desenho para um grafo com V = 10. A abordagem unificada conseguiu resultado exatamente igual ao conseguido pela abordagem híbrida, para este número de vértices.

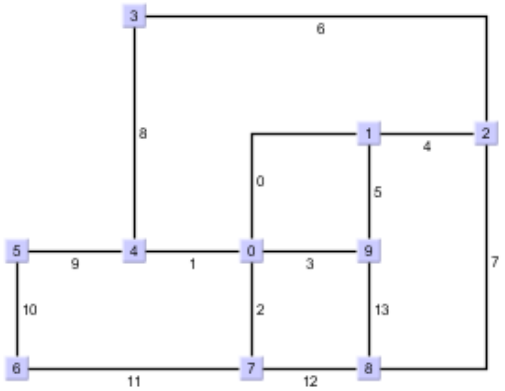


Figura 4.5 – Desenho final para o grafo com V = 10, utilizando o algoritmo genético com a soma ponderada dos objetivos na abordagem unificada.

A *Figura 4.6* mostra o desenho para um grafo com V = 20. A abordagem unificada conseguiu resultado melhor, considerando a soma total das arestas (f_L = 37), em relação ao resultado conseguido pela abordagem híbrida (f_L = 45), para este número de vértices.

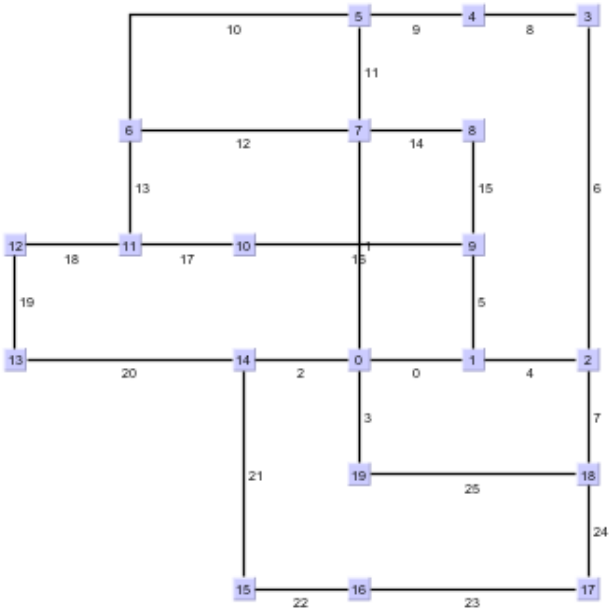


Figura 4.6 – Desenho final para o grafo com V = 20, utilizando o algoritmo genético com a soma ponderada dos objetivos na abordagem unificada.

A *Figura 4.7* mostra o desenho para um grafo com V = 30. Neste caso, a abordagem unificada conseguiu resultado levemente inferior , considerando o número de cruzamentos e o número de dobras, mas superior considerando a soma total das arestas (f_X = 5, f_B = 9 e f_L = 77), em relação ao resultado conseguido pela abordagem híbrida (f_X = 1, f_B = 5 e f_L = 85), para este número de vértices. O fato dos valores para número de cruzamentos e dobras serem inferiores podem estar relacionados com a escolha dos pesos utilizados na soma ponderada dos objetivos. Embora se tenha testado para diversos valores de pesos, esta escolha não é trivial, e por isto, não há como se garantir que a escolha feita seja a melhor possível.

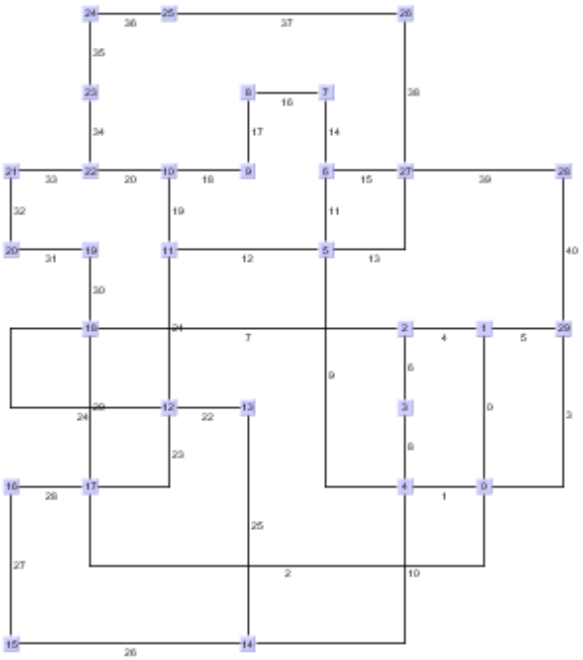


Figura 4.7 – Desenho final para o grafo com V = 30, utilizando o algoritmo genético com a soma ponderada dos objetivos na abordagem unificada.

A *Figura 4.8* mostra o desenho para um grafo com V = 40. Neste caso, a abordagem unificada conseguiu resultado inferior (f_X = 10, f_B = 10 e f_L = 159) em relação ao resultado conseguido pela abordagem híbrida (f_X = 4, f_B = 6 e f_L = 120), para este número de vértices.

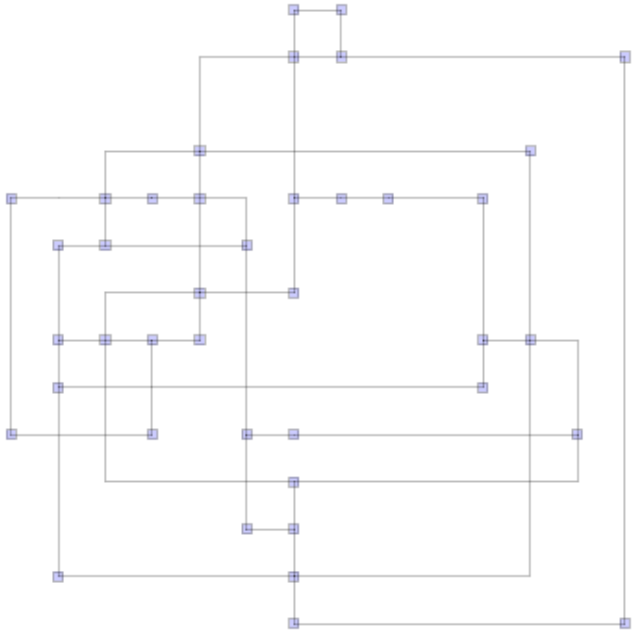


Figura 4.8 – Desenho final para o grafo com V = 40, utilizando o algoritmo genético com a soma ponderada dos objetivos na abordagem unificada.

A *Figura 4.9* mostra o desenho para um grafo com V = 50. Neste caso, a abordagem unificada conseguiu resultado inferior (f_X = 29, f_B = 22 e f_L = 337) em relação ao resultado conseguido pela abordagem híbrida (f_X = 13, f_B = 13 e f_L = 224), para este número de vértices.

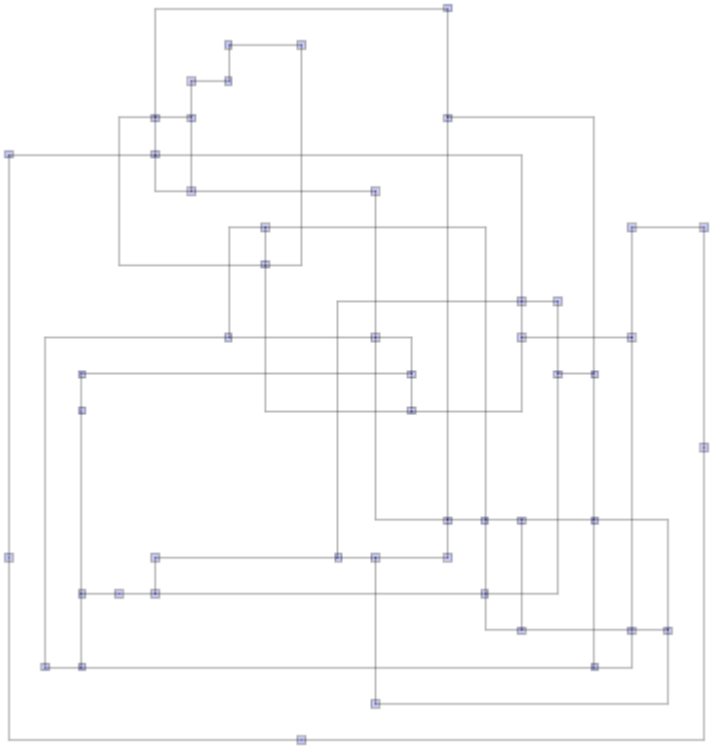


Figura .4.9 – Desenho final para o grafo com V = 50, utilizando o algoritmo genético com a soma ponderada dos objetivos na abordagem unificada.

O código desenvolvido ainda apresenta problemas de implementação e não foi possível gerar resultados com número de vértices superior a 100.

Uma análise dos resultados será feita na próxima seção.

4.6. Análise dos resultados da abordagem unificada

Os resultados obtidos pela abordagem unificada II mostram-se promissores quando comparados aos resultados da abordagem clássica e competitivos ou até melhores para alguns casos, quando comparados com os resultados das abordagens híbridas desenvolvidas. Os resultados foram muitos bons para grafos pequenos e nem tanto para grafos maiores. Para os grafos cujo número de vértices são iguais a 80 e 100, não foi possível evitar as sobreposições de arestas, o que caracterizou o desenho do grafo como de baixa qualidade. Não foi possível, também, gerar desenhos de grafos com número de vértices maior que 100, como discutido anteriormente. Além disto, o tempo de processamento por geração do algoritmo genético é maior nesta abordagem que na abordagem híbrida, sendo 25 vezes maior em média. O número de gerações requeridas para se atingir o critério de parada é consideravelmente grande nesta abordagem, o que eleva o tempo total de processamento. Embora tenha apresentado os problemas citados, esta nova metodologia permite eliminar a interdependência existente entre as etapas da *topologia-forma-métrica* e apresentou bons resultados para grafos pequenos. Além disto, ela garante flexibilidade para o tratamento dos critérios estéticos e restrições necessárias e independe das características do modelo matemático que os representam, uma vez que trabalha em conjunto com o algoritmo genético.

Há indicações de que, com a implementação de melhores heurísticas, ou modificações na estrutura de dados utilizada, resultados eficientes para grafos maiores também podem ser conseguidos. Outros operadores de busca local podem ser agregados ao processo, gerando resultados ainda melhores que os conseguidos até o momento.

Outra possível melhoria é aplicar a metodologia de tomada de decisão multicritério em ambiente *fuzzy* no processo evolucionário do AG afim de se obter soluções mais harmoniosas, o que minimiza os problemas já conhecidos quando se utiliza a abordagem por soma ponderada. Um algoritmo genético puramente multiobjetivo, como o NSGAII pode ser aplicado também nesta abordagem, o que traz a vantagem de se ter um conjunto de soluções no Pareto. Pode-se utilizar critérios estabelecidos pelos usuários para escolha de soluções no Pareto ou mesmo aplicar a técnica de tomada de decisão multicritério em ambiente *fuzzy* e escolher, no conjunto solução de Pareto, aquela que apresente uma maior harmonia.

CAPÍTULO 5

CONCLUSÕES E PROPOSIÇÕES PARA TRABALHOS FUTUROS

5.1. Conclusão

Nesta tese, foram desenvolvidas novas metodologias para desenho automático de grafos ortogonais no *grid*, baseadas em técnicas de otimização. Os algoritmos desenvolvidos trouxeram contribuições para a área de desenho de grafos, conforme mostrado nos capítulos 3 e 4. Os modelos matemáticos desenvolvidos para o tratamento do problema de compactação, utilizando técnicas de programação linear inteira e otimização multiobjetivo, geraram resultados promissores para a resolução do problema de compactação de grafos ortogonais, considerando diversos critérios estéticos como os modelados neste trabalho. Isto garante a flexibilidade desejada para o tratamento dos diversos critérios estéticos, bem como restrições. Porém, ainda permanece a necessidade do desenvolvimento de uma metodologia para a obtenção das funções objetivo de forma automática, dado um grafo *G*. A partir deste grafo *G*, seus segmentos, em função das arestas horizontais e verticais, devem ser estruturados. Então, as funções objetivo relacionadas com estes segmentos devem ser automaticamente obtidas e, desta forma, dinamizar o processo.

Os novos algoritmos desenvolvidos para a abordagem híbrida (*topologia-forma-métrica* com algoritmo genético), mostraram resultados superiores aos da abordagem clássica *topologia-forma-métrica*. O hibridismo foi considerado de três maneiras diferentes, sendo que a primeira utiliza a soma ponderada dos objetivos como função de aptidão (aptidão), no processo evolucionário do algoritmo genético. A segunda, utiliza a metodologia de tomada de decisão multicritério em ambiente *fuzzy* no processo evolucionário do algoritmo genético, garantindo, com isto, a obtenção de soluções mais harmoniosas. A terceira está relacionada com a utilização do algoritmo genético multiobjetivo, NSGAII, para a geração do conjunto solução de Pareto. No conjunto de Pareto obtido é aplicada a metodologia de tomada de decisão multicritério em ambiente *fuzzy* para a escolha da solução mais harmoniosa. Foram gerados, também, resultados para uma abordagem puramente aleatória. Neste caso, é considerada apenas a abordagem clássica e a variação aleatória da ordenação de inserção das arestas. Para tal, o problema foi formulado como um problema de busca por embutimentos de grafos planares mais otimizados, os quais são levados para as próximas etapas da abordagem clássica. Já para a abordagem híbrida, o problema é tratado como um problema de otimização combinatorial baseado em permutação e foi

resolvido utilizando o algoritmo genético. Neste caso, são considerados os operadores apropriados para a representação por permutação de inteiros. A ordem de inserção das arestas na formação dos embutimentos planares é dada pela permutação de um conjunto de inteiros, os quais representam os identificadores das arestas do grafo e são organizados de acordo com a ordem em que estas arestas são inseridas para formar tais embutimentos planares. Os testes da abordagem puramente aleatória indicaram que esta metodologia apresenta resultados intermediários entre a abordagem clássica e a abordagem híbrida, mostrando a superioridade do AG na solução do problema de desenho de grafos na abordagem híbrida.

As variações que deram origem às três abordagens híbridas estão relacionadas com o processo evolucionário do algoritmo genético, conforme mostrado. Nos três casos, o cálculo dos valores dos objetivos utiliza os algoritmos clássicos para as etapas de *planarização, ortogonalização e compactação*, levando-os a uma condição que permita calcular e considerar o número de cruzamentos e o número de dobras, bem como a soma total dos comprimentos das arestas. Foram realizadas 10 execuções do algoritmo genético para cada abordagem desenvolvida e para cada instância dos grafos de teste. Assim, dez resultados diferentes para cada instância foram obtidos em cada nova abordagem. Para escolher os resultados finais, no caso da soma ponderada dos objetivos, foi escolhido aquele que apresentou menor valor calculado da soma ponderada. Foi considerada a análise estatística (média e desvio padrão) também. Para a abordagem que utiliza tomada de decisão multicritério em ambiente *fuzzy* e para aquela que utilizada o algoritmo multiobjetivo, NSGAII, foi utilizada a tomada de decisão multicritério em ambiente *fuzzy* nos resultados obtidos nas 10 execuções do algoritmo genético para cada instância de teste. Usando os valores da função de aptidão na soma ponderada, pode observar-se que uma das duas opções para escolher os melhores resultados (melhor aptidão ou média) leva a uma situação em que os melhores resultados de média ou da função de aptidão nem sempre indicam o melhor resultado para cada objetivo que compõe a função de aptidão. Por isto, pode existir um certo desequilíbrio entre eles. Essa situação mostrou claramente a necessidade de utilizar os instrumentos decisórios para auxiliar na escolha dos resultados, considerando todos os objetivos e o grau de importância de cada um deles, se necessário. Para atingir esse objetivo, a técnica de tomada de decisão multicritério em ambiente *fuzzy* foi utilizada. Portanto, a aplicação desta metodologia no processo evolucionário ou no conjunto Pareto ótimo trouxe a garantia da escolha das soluções mais harmoniosas, ou seja, com um melhor equilíbrio na satisfação dos objetivos. Além disto, esta abordagem pode auxiliar na escolha das soluções mais harmoniosas, considerando também as 10 execuções do AG. Os resultados obtidos ilustram os benefícios das abordagens híbridas desenvolvidas, especialmente para grafos com mais de centenas de vértices e arestas. A ordem de inserção das arestas no algoritmo de planarização torna-se ainda mais relevante. O algoritmo genético é capaz de procurar por melhores ordenações de inserção destas arestas, bem como na avaliação da qualidade final do desenho. Seus resultados levam a

melhorias que podem atingir até 50% quando comparadas com os resultados da abordagem clássica, para o pior caso tratado neste trabalho.

Uma metodologia unificada para desenho automático de grafos é também desenvolvida neste trabalho. Trata-se de uma abordagem inédita para desenhos ortogonais no *grid*. A idéia inicial para o desenvolvimento desta abordagem é a de eliminar a necessidade de se tratar o desenho ortogonal de grafos em três etapas, podendo fazê-lo em apenas uma. Com isto, é possível evitar as interdependências existentes entre as etapas da *topologia-forma-métrica.* Esta metodologia foi desenvolvida com base na utilização do algoritmo genético e seus resultados iniciais mostram-se promissores.

Foram desenvolvidos operadores de busca local para auxiliar o algoritmo genético na busca por melhores resultados no espaço de busca. A aplicação do algoritmo genético na busca por soluções de maior nível de qualidade e legibilidade, em relação a satisfação dos diversos critérios estéticos tratados no problema de desenho de grafos, mostrou-se superior quando seus resultados são comparados às heurísticas clássicas, ou seja, aquelas utilizadas pela abordagem *topologia-forma-métrica*. Os resultados foram muitos bons quando são considerados grafos pequenos, chegando a ser até melhor do que os da abordagem híbrida para alguns dos casos estudados. Estes são indicativos de que com algumas melhorias na abordagem, pode-se atingir resultados iguais ou melhores aos da abordagem híbrida também para grafos maiores. Podem ser aplicadas também as técnicas de tomada de decisão multicritério em ambiente *fuzzy* nos processos evolucionários bem como o algoritmo genético multiobjetivo, NSGAII.

Conforme o que foi mostrado e discutido, pode-se observar que todos o objetivos propostos neste trabalho foram atingidos. O desenvolvimento das novas metodologias, além de oferecer técnicas mais eficientes para a solução do problema de desenho de grafos ortogonais, ainda abre caminho para novos desenvolvimentos, considerando o fato que há aproximadamente 10 anos que não se desenvolve nenhuma nova metodologia para desenhos ortogonais no *grid*. Além disto, fica claro o quão eficiente foi a utilização do algoritmo genético no tratamento de problemas de desenho de grafos ortogonais.

Os problemas remanescentes nas abordagens desenvolvidas farão parte das indicações para trabalhos futuros, listados na próxima seção.

5.2. Proposições para trabalhos futuros

- Nos estudos relacionados com a aplicação de técnicas de programação linear inteira (PLI) no problema de desenho ortogonal de grafos no *grid*:

(a) É necessário o desenvolvimento de algoritmo que permita a obtenção, de forma automática, da função objetivo de um determinado grafo, dado o seu número de vértices, arestas e a sua lista de adjacências;
(b) É necessário também o desenvolvimento de uma metodologia que melhore a aproximação linear utilizada para as funções objetivo não-lineares modeladas neste trabalho, para a aplicação da técnica de PLI;
(c) Avaliar a possibilidade de adotar outros tipos de critérios estéticos.

- Na abordagem unificada desenvolvida e apresentada neste trabalho:

(a) É importante avaliar a necessidade do desenvolvimento de algoritmos de busca local, que permitam um melhoramento nos resultados desta abordagem, de forma a torná-la mais eficiente e conpetitiva em relação a abordagem clássica (*topologia-forma-métrica*), bem como resultados mais competitivos em relação às abordagens híbridas desenvolvidas neste trabalho, quando considerados números maiores de vértices;
(b) Para o operador de busca local relacionado com a compactação, avaliar a possibilidade de aplicar técnicas de compactação unidimensional e/ou bidimensional mencionadas no capítulo 2 deste trabalho;
(c) É importante avaliar também a possibilidade de melhorias dos algoritmos desenvolvidos de forma a minimizar o tempo gasto no processamento dos grafos e o AG, seja variando o tipo de estrutura de dados a ser utilizada, seja variando o manuseio da estrutura de dados utilizada neste trabalho;
(d) Aplicar técnicas de tomada de decisão multicritério em ambiente *fuzzy* no processo evolucionário do algoritmo genético para obtenção de soluções harmoniosas;
(e) Aplicar o algoritmo multiobjetivo, NSGAII, nesta abordagem e avaliar os resultados obtidos;
(f) Corrigir o erro de implementação que não permite aplicar o algoritmo para grafos com mais de 100 vértices;
(g) Avaliar a possibilidade de adotar outros tipos de critérios estéticos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] N. GERSHON, S. G. EIKE and C. STUART. Information Visualization, **IEEE Computer Graphics and Applications**, Los Alamitos, Vol. 17, No. 4, pp. 29-31, July/Aug. 1994.

[2] T. MUNZNER. **Visualization of Large Graphs and Networks**. 2000. Tese de Doutorado – Stanford University, Stanford. Disponível em: http://graphics.stanford.edu/papers/munzner_thesis/ - [visitado em 22/10/2010].

[3] J. GROSS and J. YELLEN. **Graph Theory and its Applications**. CRC Press, New York, first edition, 1999.

[4] F. HARARY. **Graph Theory**. Addison-Wesley, first edition, 1968.

[5] J.A.BONDY and U.S.R.MURTY. **Graph Theory with Applications**, The Macmillan Press, first edition, 1976.

[6] L.T. C. MELO, **Uma Biblioteca para desenho de Grafos, Construída sobre o paradigma de programação genérica**. Dissertação de Mestrado, UFMG -Brasil, 2004.

[7] **GTAD** - Uma Biblioteca para desenho de Grafos, Construída sobre o paradigma de programação genérica: http://gtad.sourceforge.net/ [visitado em 22/10/2010].

[8] N. L. BIGGS e K. LLOYD and R. WILSON, **Graph Theory**, Oxford University Press, England, 1976.

[9] D. WEST. **Introduction to Graph Theory**. Prentice Hall, third Edition, 2006.

[10] Open Problems – **graph theory and combinatorics**. http://www.math.uiuc.edu/~west/openp/ [visitado em 22/10/2010].

[11] R. WILSON. **Introduction to Graph Theory**. Longman, Edinburgh, Harlow, England, fourth edition, 1996.

[12] K. MEHLHORN. **Data Structures and Algorithms 2**: Graph Algorithms and NPCompleteness. Springer Verlag, Germany, 1984.

[13] P. EADES and N. WORMALD. Edge crossings in drawings of bipartite graphs. **Algorithmica,** Vol. 11, No. 4, pp. 379–403, 1994.

[14] G. DI BATTISTA, R. TAMASSIA, AND I. G. TOLLIS, Area requirement and symmetry display in drawing graphs, in Proc. **ACM Symposium on Computational Geometry, Association for Computing Machinery**, New York, pp. 51-60, 1989.

[15] M. PATRIGNANI, On the Complexity of Orthogonal Compaction, **ELSEVIER - Computational Geometry :Theory and Applications**. Vol. 19, No. 1, pp. 47-67, 2001.

[16] C. BARTINI, E.NARDELLI E R. TAMASSIA, A Layout Algorithm for Data Flow Diagrams, **IEEE Transactions on Software Engineering**, Vo. 12, No. 4, pp. 538-546, 1986.

[17] R. TAMASSIA, On Embeddings a Graph in the Grid With the Minimum Number of Bends, **SIAM Journal on Computing**, Vol.16, No.3, pp. 421-444,1984.

[18] Di BATTISTA, P. EADES, R. TAMASSlA, I. G. TOLLIS, **Graph Drawing – Algorithms for the visualization Graphs**. *Prentice Hall*, 1999.

[19] R. TAMASSIA, G. Di BATTISTA and C. BATINI, Automatic Graph Drawing and Readability of Diagrams, **IEEE Transaction on Systems, Man and Cybernetic**, SMC Vol. 18, No. 1, pp. 61-79, 1988.

[20] P. BERTOLLAZI, G. Di BAPTISTA, W.DIDIMO, Computing Orthogonal Drawings with the Minimum Number of Bends, **IEEE Transaction on Computers**, Vol. 49, No. 38, pp. 826-840, 2000.

[21] G. Di BATTISTA, G. LIOTTA, F. VARGIU, Spirality and Optimal Orthogonal Drawings, **SIAM Journal on Computing**, Vol. 27, No. 6, pp.1764-1811, 1998.

[22] M. EIGLESPERGER, S.P. FEKETE, W.G. KLAU, Orthogonal Graph Drawing, **In Drawing Graphs: Methods and Models**, LNCS Vol. 2025, pp. 121-171, 2001.

[23] M. EIGLESPERGER, U. FOBMEIER, M. KAUFMANN, Orthogonal Graph Drawing with Constraints, **In Proceedings of the eleventh annual ACM-SIAM – Symposium on discrete algorithms**, California, pp. 3-11, 2000.

[24] G.W. KLAU, P. MUTZEL, Optimal Compaction of Orthogonal Grid Drawings, **Technical report**, Max-Plank Institut For Informatic, Saarbrucken, 1998.

[25] P. MUTZEL and G.W. KLAU. Optimal Compaction of Orthogonal Grid Drawings (Extended Abstract), **LNCS:Integer Programming and Combinatorial Optimization**, Vol. 1610, pp. 304-319, 1999.

[26] M. EIGLESPERGER, M. KAUFMANN, Fast Compaction for Orthogonal Drawing with Vertices of Prescribed Size, **LNCS**, No. 2265, pp. 67-71, 2002.

[27] S.I. ZUKHOVITSKIY and L.I. AVDEYEVA, **Linear and Convex Programming**, W.B. SAUNDERS COMPANY, Philadelphia, 1966.

[28] R. E. GOMORY, Outline of an algorithm for integer solutions to linear programs, **Bulletim of the American Mathmetical Society**, Vol. 64, No. 5, pp. 275-278, 1958.

[29] A. H. LAND and A. G. DOIG, An automatic method for solving discrete programming problems, **Econometrica**, Vol.28, pp. 497-520, 1960.

[30] L.A.WOLSEY, **Integer Programming**, John Willey & Sons, first edition, 1998.

[31] G.L.NEMHOUSER, L.A.WOLSEY, **Integer and Combinatorial Optimization**, John Willey & Sons, first edition, 1999.

[32] C.H. PAPADIMITRIOU, K. STEIGLITZ, **Combinatorial Optimization – Algorithm and Complexity**, Dover Publications, first edition, 1998.

[33] M.C. GOLDBARG, H.P.L. LUNA, **Otimização Combinatória e Programação Linear,** ELSEVIER, segunda edição, 2005.

[34] M. EHRGOTT, X. GANDIBLEUX, **Multiple Critéria Optimization: State of the Art Annotated Bibliographic Surveys**, Springer, first edition, 2002.

[35] M. EHRGOTT, **Multicriteria Optimization**, Springer, second edition, 2005.

[36] F. M. TERRA. Desenho de Grafos: Uma abordagem Utilizando Programação Linear Inteira. **Dissertação de Mestrado**, UFMG -Brasil, 2009.

[37] A.E. EIBEN and J.E. SMITH, **Introduction to Evolutionary Computing,** Springer**, 1st** edition, 2003.

[38] M. L. PINEDO, **Scheduling: Theory, Algorithms, and Systems,** Prentice Hall**,** 3rd Edition, 2006.

[39] P. BRUCKER, **Scheduling Algorithms**, Springer, 5th Edition, 2006.

[40] A. ABRAHAM at al, **Evolutionary Multiobjective Optimization – Theoretical Advances and Applications**, Springer, 2005.

[41] K. DEB, **Multi-Objective Optimization Using Evolutionary Algorithms**, Wiley, 2004.

[42] M. GEN and RUNWEI CHENG, **Genetic Algorithms & Engineering Optimization**. John Wiley & Sons, New York, 2000.

[43] A. GHOSH, S. DEHURI, Evolutionary Algorithms for Multi-criterion Optimization: a survey, **International Journal of Computation and Informatical Science**, Vol 2, No.1, pp. 38-54, 2004.

[44] D. E. GOLDBERG, **Genetic Algorithms in Search, Optimization and Machine Learning**, Adisson-Wesley, 1989.

[45] Z. MICHALEWICZ, **Genetic Algorithms + Data Structures = Evolution Programs.** Springer, 1998.

[46] J.H. HOLLAND, **Adaptation in Natural and Artificial Systems**, University of Michigan Press, Ann Arbor, 1975.

[47] J. BRANKE, F.BUCHER, H. SCHMECK, Using genetic algorithms for drawing undirected graphs, **University of Karlsruhe**,1998.

[48] T.C. BIELDL, B.P. MADDEN, I.G. TOLLIS, The Three-Phase Method: A Unified Approach to Orthogonal Graph Drawing, **Proceedings of the 5th International Symposium on Graph Drawing**, LNCS, Vol. 1353, pp. 391-402, 1994.

[49] R. TAMASSIA, New Layout Tecniques for Entity-Relationship Diagrams, **In Proceedings 4th International Conference on Entity-Relationship Approach,** pp. 304-311, 1985.

[50] T. LENGAUER, **Combinatorial Algorithm for Integrated Circuit Layoult**. John Wiley & Sons, New York, 1990.

[51] G. Di BAPTISTA, P. EADES, I.G. TOLLIS, R. TAMASSIA, Algorithm for Drawing Graphs: An Annotated Bibliography,**Technical Report 82**, Department of Computer Science, University of Queensland 1984.

[52] C. BATTINE, L. FURLANI, and E. NARDELLI. What's a good diagram? a pragmatic approach. **In Proceedings 4th International Conference on the Entity Relationship Approach**, 1985.

[53] H. C. PURCHASE, R. F. COHEN, and M. JAMES. Validating graph drawing aesthetics. **Graph Drawing (Proc. GD '95)**, LNCS. Vol. 1027, pp.435-446, Springer- Verlag, 1996.

[54] K. SUGIYAMA, S. TAGAWA, and M. TODA. Methods for visual understanding of hierarchical systems. **IEEE Transaction on System, Man Cybernetics**., SMC Vol. 11, No. 2: pp.109-125, 1981.

[55] R. J. LIPTON, S. C. NORTH, and J. S. SANDBERG. A method for drawing graphs. In Proc. **1st Annual ACM Symposium Computational Geometry**, pp. 153-160, 1985.

[56] C. KOSAK, J. MARKS, and S. SHIEBER. Automating the layout of network diagramas with specified visual organization. **IEEE Transation on System, Man Cybernetic**, SMC Vol.11, No.3, pp. 440-454, 1994.

[57] G. Di BATTISTA and E. NARDELLI, Hierarquies and Planarity Theory, **IEEE Transactions on Systems, MAN and Cibernetics,** Vol. 18, No. 6, November/December 1988.

[58] M. EIGLSPERGER, M.SIEBENHALLER and M. KAUFMANN, An Efficient Implementation of Sugiyama's Algorithm for Layered Graph Drawing, **Journal of Graph Algorithms and Applications**, Vol. 9, No. 3, pp. 305-325, 2005.

[59] G. Di BATTISTA and R. TAMASSlA, Algorithms for plane representations of acyclic digraphs **Theoretical Computing Science**., Vol. 61, pp. 175—198, 1988.

[60] J. HOPCROFT and E. TARJAN, Efficient planarity testing, **Journal of the Association for Computing**, pp. 549-568, 1974.

[61] G. Di BATTISTA and R. TAMASSIA. On-line planarity testing. **SIAM Journal on Computing**, Vol. 25, No. 5, pp. 956-997, 1996.

[62] J. SOUKUP. Circuit layout. **Proceedings IEEE**, Vol. 69, No. 10, pp. 197-213, 1972.

[63] H. NAGAMOCHI, T. SUZUKI, T. ISHII, A simple recognition of maximal planar graphs, **ELSEVIER– Information processing Letters**, pp.224-226, 2003.

[64] R. TAMASSIA, I.G. TOLLIS and J.S. VITTER, Lower Bounds and Parallel Algorithms for Planar Orthogonal Grid Drawings, **In Proceedings IEEE Symposium on Parallel and Distributed Processing**, pp. 386-393, 1991.

[65] W. DIDIMO, LENFORT, A. GRID: An Interactive Tool for Computing Ortogonal Drawing With the Minimum Number of Bends, **Proceedings on GD´97**, Rome, LNCS Vol.1353, pp.309-315, 1994.

[66] W. HE, K. MARRIOT. Constrained Graph Layout. In: **Internal Symposium on Graph Drawing**, GD, 4, 1996, Berkeley, California. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, LNCS Vol. 1190, pp. 217-232, 1996.

[67] G. Di BATTISTA G., GARG, A., LIOTTA, G., TAMASSIA, R. TASSINARI E., VAUGIR F., An Experimental Comparison of four Graph Drawing Algorithms, **11th Annual ACM Symposium on Computational Geometry**, Vancouver, 1995.

[68] C. KURATOWSKI, Sur le probleme des corbes gauches en topologie, **Fundamenta Mathematicae** Vol. 15, pp.271-283, 1930.

[69] L. AUSLANDER and S.V. PARTER, On embeddings graphs in the plane, **J. Math. and Mech**, Vol. 10, No. 3, 517-523,1961.

[70] A. J. GOLDSTEIN, An efficient and constructive algorithm for testing whether a graph can be embedded in a plane, **Graph and Combinatorics Conference**, Dept. Math., Princeton University, pp. 16-18, 1963.

[71] A. LEMPEL, S. EVEN and I. CEDERBAUM, An algorithm for planarity testing of graphs, **Proceedings International Symposium on Theory of Graphs** (New York) (P. Rosenstiehl, ed.), Gordon and Breach, pp. 215-232, July 1964.

[72] R.E. TARJAN, Implementation of an efficient algorithm for planarity testing of graphs, Dec. 1969.

[73] J. HOPCROFT, and R. TARJAN. Planarity testing in V log V steps: Extended abstract. Proc. IFIP Cong. 1971. **Foundations of Information Processing, Ljubljana**, Yugoslavia, North-Holland Pub. Co., Amsterdam, pp. 18-22, Aug. 1971.

[74] K. MEHLHORN and P. MUTZEL, On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm, **Algorithmica,** Vol. 16, No. 2, pp.233-242, 1996.

[75] P. MUTZEL, A fast embedding algorithm based on the Hopcroft-Tarjan planarity test, **Technical Report 92.107**, Universität zu Köln, 1992,
http://www.mpi-sb.mpg.de/~ mutzel/mpireports/publications.html.

[76] K.S. BOOT and G.S. LUEKER, Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, **Journal of Computer and System Science** No.13, pp. 335-379, 1976.

[77] W. K. SHIH e W.L. HSU, A new planarity test, **Theoretical Computer Science,** No. 223, pp. 179-191, 1999.

[78] J. BOYER and W. MYRVOLD, Stop minding your P's and Q's: A simplified planar embedding algorithm, **Proceeding of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms** (Baltimore, Maryland), ACM Special Interest Group on Algorithms and Computation Theory and SIAM Activity Group on Discrete Mathematics, ACM Press, pp. 140-146, January1999.

[79] E.R. CANFIELD and S.G. WILLIAMSON, The two basic linear time planarity algorithms: Are they the same?, **Linear and Multilinear Algebra,** No**.** 26, pp.243-265, 1990.

[80] R. TAMASSIA. A dynamic data structure for planar graph embedding, **in Automata, Languages and Programming (Proceedings 15th International Colloquium on Automata, Languages and Programming)**, LNCS Vol. 317, Springer-Verlag, Berlin, New York, Heidelberg, pp. 576-590, 1988.

[81] G. Di BATTISTA and R. TAMASSIA. On-line planarity testing. **SIAM Journal on Computing**, Vol. 25, No. 5, 956-997, 1996.

[82] G. Di BATTISTA and R. TAMASSIA, Incremental planarity testing. In **Proceedings of the 30th Symposium on the Foundations of Computer Science (FOCS'89)**, pp. 436-441, 1989.

[83] M. JUNGER and P. MUTZEL, Maximum Planar Subgraphs and Nice Embeddings: Practical layout Tools, **Algorítmica**, Vol. 16, pp. 33-59, 1996.

[84] A. GARG, R. TAMASSIA, A New Minimum Cost Flow Algorithm with Applications to Graph Drawing, **in S.C.North editor, Graph Drawing (Proc.GD'96), LNCS.** Spring-Verlag, 1994.

[85] A. PAPAKOSTAS, I.G.TOLLIS, Orthogonal Drawing of High Degree Graphs with Small area and few bends, **LNCS**, Vol.1272, pp. 354-357, 1994.

[86] G. Di BATTISTA, A. GARG, G. LIOTTA, R. TAMASSIA, E. TASSINARI and F. VAUGIR, An Experimental Comparison of three Graph Drawing Algorithms, **Computational Geometry: Theory and Applications**, 1996.

[87] U. FOBMEIER, M. KAUFMANN, Drawing High Degree Graphs with Low Bends Numbers, **Proceedings of GD´95**, Passau, LNCS Vol. 1027, pp. 254-266, 1995.

[88] F. P. PREPARATA and M. I. SHAMOS, **Computational Geometry – an Introduction,** Springer Verlag, 1985.

[89] **S.** SCHAIBLE and T. IBARAKI, Fractional Programming, **European Journal of Operational Research**, Vol. 12, pp. 325-338, 1983.

[90] C. SINGH, Optimality Conditions in Fractional Programming, **Journal of Optimization Theory and Applications**, Vol. 33, No. 2, 1983.

[91] http://lpsolve.sourceforge.net/5.5/ [visitado em 22/10/2010].

[92] R. SEDGEWICK. **Algorithms in C++ - Fundamentals, Data Structures, Sorting and Searching**. Addison-Wesley, USA, third edition, 1998.

[93] R. SEDGEWICK. **Algorithms in C++ - Graph Algorithms.** Addison-Wesley, USA, third edition, 2002.

[94] D. KLOBER and A. G. B. TETTAMANZI, Recombination Operators for Evolutionary Graph Drawing, **Proceedings of Parallel Problem Solving from Nature PPSN-V**, LNCS, Vol. 1498, pp. 988-997, 2006.

[95] R. CIMIKOWSKI, An analysis of some linear graph layout heuristics, **Journal of Heurístics,** Vol. 12, No.3, pp. 143-153, 2006.

[96] P. KUNTZ, B. PINAUD, R. LEHN, Minimizing crossings in hierarchical digraphs with a hybridized genetic algorithm, **Journal of Heurístics,** Vol. 12, No.1-2, pp. 23-36, 2006.

[97] M. LAGUNA, R. MARTI, V. VALLS, Arc Crossing Minimization in Hierarchical Digraphs with Tabu Search, **Elsevier Science**, Vol. 24, No. 12, pp. 1175-1186, 1997.

[98] A. O. RODRIGUÉZ, A. R. SUÁREZ, Genetic Graph Drawing, 13th International Conference AIENG'98, pp. 1-17, 1998.

[99] A.R. SUAREZ, M. SEBAG and A.O. RODRIGUEZ, A study of Evolutionary Graph Drawing, **Technical Report**, Sep 1999 .

[100] A. M. S. BARRETO and H. J. C. BARBOSA. Graph layout using a genetic algorithm. **In VI Brazilian Symposium on Neural Networks** (SBRN`00), pp. 179, 2000.

[101] A. R. SUAREZ, A. O.RODRIGUEZ, and M. SEBAG. Automatic graph drawing and stochastic hill climbing. **In Proceedings of Genetic and Evolutionary Computation Conference (GECCO 99),** ACM Press, pp. 1699–1706, 1999.

[102] D. VRAJITORU. Hybrid multiobjective optimization genetic algorithms for graph drawing. **In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 07)**, ACM Press, pp. 912–912, 2007.

[103] J. W. GUDENBERG, A. NIEDERLE, M. EBNER, and H. EICHELBERGER. Evolutionary layout of UML class diagrams. **In Proceedings of the 2006 ACM symposium on Software visualization,** pp. 163–164, 2006.

[104] H. A. D. DO NASCIMENTO. A Focus and Constraint-Based Genetic Algorithm for Interactive Directed Graph Drawing. **In: Second International Conference of Hybrid Intelligent Systems**, 2002, Santiago. Soft Computing Systems - Design, Management and Applications. Amsterdam : IOS Press. Vol. 87, pp. 634-643, 2002.

[105] A. A. Lyapunov, Operational research. methodological aspects (in russian), Nauka, Moscou, 1972.

[106] P.Ya. EKEL and E.A. GALPERIN, Box-triangular multiobjective linear programs for resource allocation with application to load management and energy market problems, **Mathematical and Computer Modelling**, Vol. 37, No. 1, pp. 1-17, 2003.

[107] P. Ya. EKEL, M. MENEZES, and F. SCHUFFNER NETO, Decision making in fuzzy environment and its application to power engineering problems, **Nonlinear Analysis: Hybrid Systems,** Vol. 1, No. 4, pp. 527-536, 2007

[108] L. CANHA, P. EKEL, J. QUEIROZ, and F. SCHUFFNER NETO, Models and methods of decision making in fuzzy environment and their applications to power engineering problems, **Numerical Linear Algebra with Applications**, Vol. 14, No. 3, pp. 369-390, 2007.

[109] P.Ya. EKEL, Methods of decision making in fuzzy environment and their applications, **Nonlinear Analysis: Theory, Methods and Applications** Vol. 47, No. 5. pp. 979-990, 2001.

[110] P.Ya. EKEL, Fuzzy sets and models of decisions making, **Computer and Mathematics with Applications**, Vol. 44, No. 7, pp. 863-875, 2002.

[111] P.Ia. EKEL, R.L.J. CARNEVALLI, B.M. MENDONÇA NETA, D.L. SOUZA, R.M. PALHARES, Modelos e métodos de tomada de decisões em ambiente fuzzy e suas aplicações**,** em **Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional**. Natal, pp. 980-990, 2003.

[112] R. E. BELLMAN & L.A.ZADEH, Decision-making in a fuzzy environment, **Management Science**, Vol 17, No. 4, pp. B141-B164, 1970.

[113] W. PEDRYCZ, P. EKEL, and R. PARREIRAS, Fuzzy Multicriteria Decision-Making: Models, Methods, and Applications, John Wiley & Sons, In Press, 2010.

[114] A. KONAK, D.W. COIT , A.E. SMITH , Multi-Objective Optimization Using Genetic Algoritms: A Tutorial, **Journal Reliability Engineering & System Safety**, Vol. 91, No. 9, pp. 992-1007, 2006.

[115] D. F. JONES, S.K. MIRRASAVI, M. TAMIZ, Multiobjective Meta-heuristics: an overview of the current state-of-the-art, **European Journal of Operational Research** , Vol. 137, No. 1, 2002.

[116] J.D. SCHAFFER, Multiple Objective optimization with vector evaluated genetic algorithms, **in International Conference on Genetic Algorithm and their applications**, L. Erlbaum Associates Inc, pp. 93-100, 1985.

[117] C.M. FONSECA and P.J. FLEMING, Multiobjective genetic algorithms, **in IEE Colloquium on Genetic Algorithms for Control Systems Engineering,** Digest, No. 1993/130, 1993, London, UK:IEE.

[118] T. MURATA and H. ISHIBUCHI, Multi-objective genetic algorithms and its applications to flowshop scheduling, **Computers & Industrial Engineering,** Vol. 30, No 4, pp 957-968, 1996.

[119] N. SRINIVAS and K. DEB, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, **Journal of Evolutionary Computation**, Vol. 2, No. 3, pp. 221-248, 1994.

[120] E. ZITZLER, and L. THIELE, Multiobjective Evolutionary Algorithms: a comparative case study and the strength Pareto approach, **IEEE Transactions on Evolutionary Computation, Vol.** 3, No. 4, pp. 257-271, 1999.

[121] J.D. KNOWLES and D.W. CORNE, Approximating the nondominated front using the Pareto archived evolution strategy, **Evolutionary Computation**, Vol. 8, No. 2, pp, 149-172, 1999.

[122] R. SARKER, K.H. LIANG and C. NEWTON, A new multiobjective evolutionary algorithm, **European Journal of Operational Research**, Vol. 140, No. 1, pp. 12-23, 2002.

[123] H. LU and G.G. YEN, Rank-desnity-based multiobjetive genetic algorithm and benchmark test function study, **IEEE Transaction on Evolutionary Computation,** Vol. 7, No. 4, pp. 325-343, 2003.

[124] K. DEB, A. PRATAP, S. AGARWAL and T. MEYARIVAN, A fast and elitist multiobjective genetic algorithm: NSGAII, **IEEE Transactions on Evolutionary Computation**, Vol. 6, No.2, pp. 182-197, 2002.

[125] B. M. M NETA, G. H. D. ARAUJO, F. G. GUIMARÃES and R. C. MESQUITA, A hybrid genetic algorithm for automatic graph drawing based on the topology-shape-metric approach, In **Proceedings of the Genetic and evolutionary Computation Conference (GECCO 2010),** ACM Press, pp. 743–750, 2010.

[126] B. M. M. NETA, G. H. D. ARAUJO, F. G. GUIMARÃES, R. C. MESQUITA and P. Ya. EKEL, A Fuzzy Genetic Algorithm for Automatic Orthogonal Graph Drawing, submetido ao **Jounal of Applied Soft Computing-ELSEVIER** em setembro de 2010, aguardando resultado.

[127] H. C. PURCHASE. Effective information visualisation: a study of graph drawing aesthetics and algorithms. **Journal of Interacting with Computers - ELSEVIER**, No. 13, PP. 147-162, 2000.

ANEXO I

ALGORITMOS QUE COMPÕEM A ABORDAGEM UNIFICADA

Primeiramente são listados os algoritmos desenvolvidos. Os mesmos serão discutidos e descritos na sequência.

Listas dos algoritmos desenvolvidos:

1. Algoritmo AG-DG Unificado //Principal
2. Algoritmo Gera_população_inicial
3. Algoritmo Seleção_torneio_binário
4. Algoritmo Cruzamento
5. Algoritmo Mutação
6. Algoritmo Rand_trig
7. Algoritmo Limpa_dados
8. Algoritmo Seta_mapa_de_inserção
9. Algoritmo Seta_arestas
10. Algoritmo Polarização_arestas
11. Algoritmo Polarização_deltas
12. Algoritmo Calcula_fitness
13. Algoritmo Matriz_checagem
14. Algoritmo de Compactação;
15. Algoritmo de Melhoria_local;
16. Algoritmo Loop_reajuste.

Algoritmo AG-DG Unificado: algoritmo principal da abordagem unificada, é por onde são feitas as chamadas aos demais algoritmos e a geração dos arquivos que armazenam os resultados.

Algoritmo Gera_população_inicial: algoritmo responsável pela geração dos novos genomas de forma totalmente aleatória. Os genomas gerados compõem a população inicial.

Algoritmos Seleção_torneio_binário, Cruzamento e Mutação: são os algoritmos que compõem o processo evolucionário do algoritmo genético.

Algoritmos Rand_trig, Limpa_dados e Calcula_fitness: são os algoritmos que auxiliam durante a execução do algoritmo genético. O algoritmo **Rand_trig** calcula os valores das perturbações, com base na distribuição Gaussiana para as coordenadas de vértices e os valores dos Δ_1. O algoritmo **Limpa_dados** é responsável pela limpeza das estruturas de dados, quando esta se faz necessário. O algoritmo **Calcula_fitness**, é responsável pelo cálculo do valor da função de aptidão, sempre que um novo individuo é gerado.

Algoritmo Seta_mapa_de_inserção: algoritmo responsável pela geração aleatória do grafo esparso, de acordo com o número definido de vértices definido. Sua saída é a estrutura de dados com as informações do grafo.

Algoritmo Seta_arestas: algoritmo responsável por inserir cada aresta do grafo na **Matriz_checagem** de acordo com as informações que compõem cada aresta do grafo.

Algoritmo Polarização_arestas: algoritmo responsável por provocar melhorias nas configurações de 20% das arestas que apresentam um maior número de sopreposições, de forma a minimizar o número destas. Assim, os indivíduos se tornam mais competitivos no processo evolucionário.

Algoritmo Polarização_deltas: algoritmo responsável por provocar melhorias nas configurações dos deltas que apresentam um tamanho de aresta muito grande, de forma a minimizar estes tamanhos.

Algoritmo Matriz_checagem: algoritmo responsável por embutir o grafo no grid e calcular o número de cruzamentos, o número de sobreposições, o número de dobras e a soma total das arestas do grafo.

` **Algoritmo de Compactação :** algoritmo que auxilia o algoritmo genético na compactação do desenho, fazendo uma ordenação nas coordenadas dos vértices, tanto na horizontal quanto na vertical e movimentando os mesmos para baixo, cima, direita ou esquerda, quando houver possibilidades de melhorias.

Algoritmo de Melhoria_local : Provoca uma perturbação de uma unidade nas coordenadas dos vértices, em todas as direções, mantendo sempre aquela que provoca melhorias no grafo *G*.

Algoritmo Loop_reajuste: algoritmo responsável por provocar melhorias nos melhores indivíduos após a conversão do algoritmo genético, ou seja, provoca mutações controladas nas arestas dos melhores indivíduos e avalia na Matriz de checagem se houve melhorias efetivas no individuo. Se sim, o mantém; senão, continua com o anterior.

A seguir serão apresentados os algoritmos descritos anteriormente:

***Algoritmo* AG-DG Unificado**

Entrada: número de vértices do graph *G* (**NVertices**), tamanho da população (**PopSize**), variável para controle de convergência (***loops***), pesos para as funções objetivo ($\alpha_1, \alpha_2, \alpha_3, \alpha_4$ e α_5):
$f_{SEV} + f_{SE}$ = Número de sobreposições entre (arestas-vértices + arestas-arestas)-(α_1);
f_{SV} = Número de sobreposições entre vértices(α_2);
f_X = Número de cruzamentos entre arestas (α_3);
f_B = Número de dobras (α_4);
f_L = soma total dos tamanhos das arestas(α_5);

Saida: *O*s valores para ($f_{SEV} + f_{SE}$), f_{SV}, f_X, f_B, f_L *e* ***F.*** *e o* desenho ortogonal otimizado;

```
Inicia-cronometro;
PopIni = Gera_população_inicial (grafo G, PopSize); //População inicial é
gerada randomicamente e apenas uma vez.
Enquanto (!critério de parada)
{
    PopSelected = Seleção_torneio_binário (PopIni);
    PopCrossed = Cruzamento (PopSelected);
    PopMutated = Mutação (PopCrossed);
    PopOffSpring = PopMutated;
    Para cada (individuo na PopOffSpring) //necessário para fazer a matriz de
                checagem  com os novos valores dos novos indivíduos
    {
        Limpa_dados (PopIni, PopSelected, PopCrossed, PopMutated,
                    Variáveis);
        Para cada (aresta e ∈ E do novo individuo)
        {
            Seta-arestas (PopOffSpring);
        }
        //com chance de 1/15, este individuo aciona a função de polarização
        Polarização (genoma g) // busca local
        //com chance de 1/10, este individuo aciona a função de melhoria local
        Melhoria_local(genoma g);
        Compactação(genoma g);
```

Calcula_fitness (genoma **g);**
Se (a aptidão do individuo atual é melhor que aptidão do melhor indivíduo armazenado**)**
{
MelhorInd = genoma **g** //armazena o melhor individuo encontrado;
}
}
PopIni = PopOffSpring; //população atual recebe os novos indivíduos para repetir processo.
Soma-se os valores de fitness de todos os indivíduos de PopOffSpring e divide por NPop, guardando o resultado = média da geração (é utilizado pelo usuário para avaliar a convergência através de um gráfico).
Loop_ control += 1;
Se (loop_control = = loops)
{
Gera-se um arquivo .gml com as informações do grafo até o momento e um arquivo .txt com as características do grafo geradas até então; OBS: é usado para parar o processo de forma controlada pelo usuário, se desejado.
}
}
Loop_reajuste(genoma **g**);

***Algoritmo* Gera_população_inicial**

Entrada: Graph *G* (**NVertices**), Tamanho da população (**PopSize**),
Saida: População inicial;
//Cria-se um genoma e atribui-se as características do individuo a este genoma;
Para (i = 0 até i < **PopSize**)
{
Limpa_dados (genoma ***g***); //limpa os dados do genoma anterior
gera_novo_genoma_aleatoriamente (genoma ***g***);
{
Para (cada vértice $v \in V$, do grafo *G*)
{
Atribui-se os valores para as coordenadas (x, y) do vértice $v \in V$, aleatoriamente, com valores entre 0 e $2 \bullet V$;
}

```
            Para (cada aresta e ∈ E do grafo G)
            {
                        Atribui os valores para cada bit no vetor de bits,
                        por exemplo ( 0 0 0);
                        O primeiro bit é escolhido aleatoriamente  e os valores para
                        Δ1 e Δ2 são pré-definidos e iguais a 0 e  0;
                        Seta_arestas (aresta e) //constroe-se a matriz de checagem
                                                    para a aresta e;
            }
            Calcula_fitness;
            Insere novo genoma (g) em PopIni ;
        }
}
```

Algoritmo = **Seleção_torneio_binário**

Entrada: População inicial (**PopIni**),
Saida: Populaçao selecionada (**PopSelected**);

```
PopAux = PopIni;
Ordena PopAux por valores de aptidão;

Para (i = 0 até i < tamanho de PopAux)
{
    Para (i = 0 até i < 2) //faz-se a seleção aleatória por duas vezes
    {
        Seleciona aleatoriamente 2 posições de PopAux;
        PopAux1 = elementos selecionados; //insere os elementos selecionados em
        PopAux1.
    }
    Ordena PopAux1 por valores de aptidão;
    Seleciona o indivíduo de menor valor de aptidão em PopAux1 e atribui a
    PopSelected;
    Limpa PopAux1;
}
```

***Algoritmo* Cruzamento**

Entrada: População selecionada (**PopSelected**),
Saida: População após cruzamento (**PopCrossed**);

Para (cada par de indivíduos de **PopSelected**)
{
Insere diretamente 20% dos individuos em **PopCrossed1** sem sofrerem cruzamento;
Para (os demais indivíduos (80%))
{
PopCrossed1 recebe 35% dos indivíduos os quais sofrem
Cruzamento_vertices_médias (par de individuos**)**;
{
Para (cada indivíduo de **PopSelected**)
{
Para (cada vértice do indivíduo)
{

$$x_{offspring1} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai1} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai2};$$

$$x_{offspring2} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai2} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet x_{pai1};$$

$$y_{offspring1} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai1} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai2};$$

$$y_{offspring2} = (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai2} + (1 - (\psi \bullet (2\varepsilon + 1) - \varepsilon) \bullet y_{pai1};$$

ψ é um número aleatório entre 0 e 1 e ε = 20% (0.2), corresponde ao Valor que se deseja aumentar na área de busca do cruzamento;
}
}
}

PopCrossed1 recebe 65% dos indivíduos os quais sofrem
Cruzamento_discreto_vertices (par de individuos**)**;
{
Para (cada indivíduo de **PopSelected**)
{
Seleciona aleatoriamente dois pontos de corte no mapa de coordenadas de vértices. Os vértices, neste intervalo, recebem coordenadas do pai_1 e fora dele, recebem coordenadas do pai_2. Depois o processo se inverte: os vértices neste mesmo intervalo, do segundo filho, recebem as coordenadas do pai_2 e fora dele, as coordenadas do pai_1.

```
                }
        }
    }
// toda a população sofre Cruzamentos_arestas
Cruzamentos_arestas (par de individuos)
{
        Para (cada par de indivíduos de PopCrossed1)
        {
                Para Δ1 e Δ2:
                {
                        Δ1 offspring1 = (ψ •(2ε +1) - ε )• Δ1 pai1 + (1–(ψ •(2ε +1) - ε ) • Δ1 pai2;
                        Δ1 offspring2 = (ψ •(2ε +1) - ε )• Δ1 pai2 + (1–(ψ •(2ε +1) - ε ) • Δ1 pai1
                        Δ2 offspring1 = (ψ •(2ε +1) - ε )• Δ2 pai1 + (1–(ψ •(2ε +1) - ε ) • Δ2 pai2;
                        Δ2 offspring2 = (ψ •(2ε +1) - ε )• Δ2 pai2 + (1–(ψ •(2ε +1) - ε ) • Δ2 pai1
                `       ε → 20% = 0.2.
                }

                Para o vetor de bits:
                {
                        Seleciona-se um único ponto de corte no vetor de bits,
                        aleatoriamente. O primeiro filho recebe os bits do pai1 antes do
                        ponto de corte e os bits do pai2, depois do ponto de corte. O segundo
                        filho recebe os bits do pai2 antes do ponto de corte e os bits do pai1,
                        depois do ponto de corte.

                        Insere indivíduos cruzados em PopCrossed;
                }
        }
 }
}
```

***Algoritmo* Mutação**

Entrada: população após cruzamentos (**PopCrossed**),
Saida: população após mutação (**PopOffSpring**);

Embaralha os elementos da população de PopCrossed e seleciona aleatoriamente 10% da população para sofrer **Mutação_vertices**;

Mutação_vertices (PopCrossed)

{

Cada vértice do individuo a ser mutado, terá 50% de chance de ser mutado

Se (vértice *v* for selecionado para mutação)

{

x' = *x* + **Rand_trig** (μ, σ); //mutaçao Gaussiana

y' = *y* + **Rand_trig** (μ, σ); //mutaçao Gaussiana

Se (o novo vértice *v* (*x', y'*) está dentro do limite do grid da matriz de checagem)

então o vértice mantém tal mutação;

Insere individuo mutado em **PopOffSpring**;

}

}

Mutação_arestas (PopCrossed)

{

Cada aresta do individuo a ser mutado terá 50% de chance de ser mutada

Se (aresta *e* for selecionada para mutação)

{

Δ_1 += **Rand_trig** (μ, σ) ;

Δ_2 += **Rand_trig** (μ, σ) ;

Um bit do vetor de bits, escolhido aleatoriamente, é invertido;

Insere individuo mutado em **PopOffSpring**;

}

}

***Algoritmo* Rand_trig**

Entrada: μ, σ

Saida: valor da perturbação nas coordenadas do vértice e nos valores de deltas;

//distribuição gaussiana para calculo dos desvios.

desvioDisponivel = falso;

Se (!**desvioDisponivel**)

{

distancia = sqrt(-2.0 • log(double(rand()) / double(RAND_MAX)));

anglo = 2.0 • PI • (double(rand()) / double(RAND_MAX));

```
        //calcula e armazena o primeiro desvio e seta a flag desvioDisponivel
        desvio = distancia•cos(anglo);
        desvioDisponivel = verdadeiro;

        //calcula e retorna o segundo desvio
        se((distancia • sin(anglo) • σ + μ > 1)
            retorna ((2-(distancia • sin(anglo) • σ + μ))• (1/ σ ));
        retorna ((distancia • sin(anglo) • σ + μ ) • (1/ σ ));
}
```

Se o desvio está disponível a partir da chamada anterior a esta função, ele é retornado, e a flag **desvioDisponivel** é setada para falso.

```
senão
{
    desvioDisponivel = falso;
      se ((desvio • σ + μ ) > 1)
          retorna ((2 - (desvio • σ + μ )) • (1/ σ ));
      retorna ((desvio • σ + μ ) • (1/ σ ));
}
```

***Algoritmo* Limpa_dados**

Entrada: genoma *g*
Saida: estrutura de dados zeradas;

Zerar as estruturas de dados e variáveis utilizadas no passo anterior;

Incrementa-se de uma unidade, o **id** do indivíduo, pois, pressupõe-se que quando um genoma é zerado é porque, outro novo ocupará o seu lugar;

***Algoritmo* Seta_mapa_de_inserção**

Entrada: **NVertices** do grafo *G* a ser gerado,
Saida: Grafo esparso gerado aleatoriamente, número de dobras;

Gera o grafo esparso aleatoriamente e retorna a estrutura de dados chamada mapa de inserção, onde contém informações sobre cada aresta, seu vértice de origem e de destino;

***Algoritmo* Seta_arestas**

Entrada: **id** da aresta;
Saida: aresta setada na matriz de checagem;

Armazena-se as coordenadas $C_v(x, y)$ do vértice de origem e de destino da aresta a ser adicionada na matriz de checagem;
Os bits do vetor de bits [0 0 1] são descritos assim: o primeiro bit representa a direção (*d*) do primeiro passo da aresta em análise: 0 se vertical e 1 se horizontal. O segundo bit representa se Δ_1 será utilizado no passo ou não: se Δ_1 for igual a 0 não o será e se Δ_1 for igual a 1, o será. O terceiro bit representa se Δ_2 será utilizado no passo ou não: se Δ_2 for igual a 0 não o será e se Δ_2 for igual a 1, o será.

Primeiro Caso (*d* 0 0): // Neste caso, como os valores de $\Delta_1 = 0$ e $\Delta_2 = 0$, ambos são desconsiderados. Os vértices de origem e destino são ligados pelo menor caminho possível: se estiverem alinhados → zero dobras, se não → 1 dobra. Assim, Δ_3 e Δ_4 são utilizados, de acordo com as equações (4.3) e (4.4).

Se *d* = 0, a aresta sairá verticalmente:
{
O próximo y (next y) → Ny = target de y;
M**atriz_checagem(Ny);**
Se (Ny != y)
Moved_y = true;

O próximo x (next x) → Nx = target de x;
Matriz_checagem(Nx);
Se (Nx != x e moved_y) //configura a dobra
Ndobras ++;
}

Se *d* = 1, a aresta sairá horizontalmente:
{
O próximo x (next x) → Nx = target de x;
Matriz_checagem(Nx);

Se (Nx != x)
Moved_x = true;

O próximo x (next x) → Nx = target de x;
Matriz_checagem(Nx);
Se (Ny != y e moved_x) //configura a dobra
Ndobras ++;
}

Segundo Caso (*d* 1 0) → Δ_1 é considerado ou *d* 0 1 → Δ_2 é considerado**):**

Se *d* = 0, a aresta sairá verticalmente:
{
Ny = Δ_1;
Matriz_checagem(Ny);
Se (y != Ny)
Moved_y = true;

Nx = target de x;
Matriz_checagem(Nx);
Se (Nx != x e moved_y) //configura a dobra
{
Ndobras ++;
moved_x = true;
moved_y = false;
}

Ny = target de y;
Matriz_checagem(Ny);
Se (Ny != y e moved_x) //configura a dobra
Ndobras ++;
}

Se *d* = 1, a aresta sairá horizontalmente:
{
Nx = Δ_1;
Matriz_checagem(Nx);
Se (x != Nx)
Moved_x = true;

Ny = target de y;
Matriz_checagem(Ny);
Se (Ny != y e moved_x) //configura a dobra

```
        {
                Ndobras ++;
                moved_y = true;
                moved_x = false;
        }

        Nx = target de x;
        Matriz_checagem(Nx);
        Se (Nx != x  e moved_y ) //configura a dobra
          Ndobras ++;
}
```

Terceiro Caso (*d* 1 1 → Δ_1 e Δ_2 são considerados**):**

Se *d* = 0, a aresta sairá verticalmente:

```
{
        Ny = Δ1;
        Nx = Δ2;
        Matriz_checagem(Ny);
        Se (y != Ny)
          Moved_y = true;
        Matriz_chacagem(Nx)
        Se (Nx != x  e moved_y ) //configura a dobra
        {
                Ndobras ++;
                moved_x = true;
                moved_y = false;
        }

        Ny = target de y;
        Matriz_checagem(Ny);
        Se (Ny != y  e moved_x ) //configura a dobra
        {
                Ndobras ++;
                Moved_x = false;
                Moved_y = true;
        }
        Nx = target de x;
        Matriz_checagem(Nx);
        Se (Nx != x  e moved_y ) //configura a dobra
```

```
            {
                    Ndobras ++;
                    moved_x = true;
                    moved_y = false;
            }
    }
```

Se *d* = 1, a aresta sairá horizontalmente:

```
    {
            Nx = Δ1 ;
            Ny = Δ2;
            Matriz_checagem(Nx);

            Se (x!= Nx)
              Moved_x = true;
            Matriz_chacagem(Ny)
            Se (Ny != y  e moved_x ) //configura a dobra
            {
                    Ndobras ++;
                    moved_y = true;
                    moved_x = false;
            }
            Nx = target de y;
            Matriz_checagem(Nx);
            Se (Nx != y  e moved_y ) //configura a dobra
           {
                    Ndobras ++;
                    Moved_y = false;
                    Moved_x = true;
            }
            Ny = target de y;
            Matriz_checagem(Ny);
            Se (Ny != y  e moved_x ) //configura a dobra
            {
                    Ndobras ++;
                    moved_y = true;
                    moved_x = false;
            }
     }
}
```

***Algoritmo* Polarização_arestas**

Entrada: individuo a ser polarizado;
Saida: individuo polarizado;

Ordenar as arestas do individuo em ordem decrescente em relação ao número de sobreposições.
Os 20% das arestas que apresentarem mais sobreposições serão polarizadas;
Para (cada aresta a ser polarizada**)**
{
Testa se a aresta está saindo ou chegando junto de outra aresta no vértice.
Caso positivo:
{
Verificar na matriz de checagem onde, em torno do vértice, tem um espaço vazio e fazer com que a aresta saia por este espaço vazio;
}
Caso negativo:
{
Mudar as características da aresta de forma a evitar a superposição
{
33% de chance mudando qualquer bit de seu vetor de bits;
33% de chance mudando o valor de Δ_1;
33% de chance mudando o valor de Δ_2;
}
}
}

***Algoritmo* Polarização_deltas**

Entrada: individuo a ser polarizado em delta;
Saida: individuo polarizado em delta;

Para (cada aresta do desenho a ser polarizada**)**
{
Se (bitvector = *d*01)
{
testar todos os pontos do grid (0-2V) para Δ_2 e manter o que gerar melhor

aptidão
}

Se (bitvector = *d*10)
{
testar todos os pontos do grid (0-2V) para Δ_1 e manter o que gerar melhor aptidão
}

Se (bitvector = *d*11)
{
testar todos os pontos do grid (0-2V) para Δ_1 e Δ_2 e manter o que gerar melhor aptidão
}
}

***Algoritmo* Calcula_fitness**

Entrada: f_{SE}, f_{SEV}, f_{SV}, $f_{X.}$, f_B *e* f_L, α_1, α_2, α_3, α_4, α_5.

Saida: valor de ***F***;

$$\boldsymbol{F} = (\alpha_1 \bullet (f_{SE} + f_{SEV}) + \alpha_2 \bullet f_{SV} + \alpha_3 \bullet f_X + \alpha_4 \bullet f_B + \alpha_5 \bullet f_L)$$

***Algoritmo* Matriz_Checagem**

Entrada: $C_v(x_i, y_i)$, e_i, N_x , N_y

Saida: Número de cruzamentos, número de sobreposições (arestas, arestas-vertices e vértices), soma total das arestas;

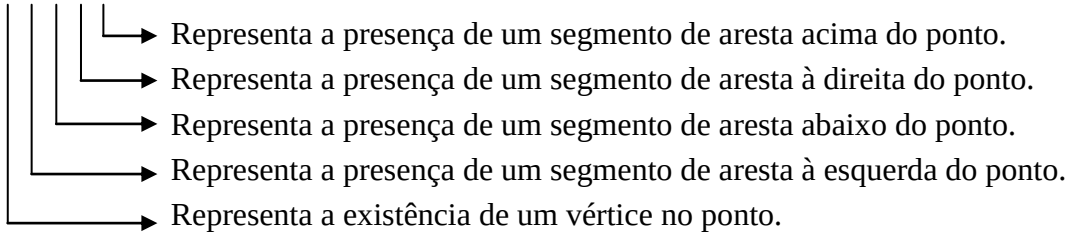
{

Percorrer ponto a ponto da matriz de inteiros, partindo de *x* ou *y* indo em direção a *Nx* ou *Ny*;

Cada valor na matriz representa as coordenadas *x* e *y* de um ponto que pode conter um vértice, um segmento de aresta ou ambos, com informações sobre a sua vizinhança e que são representadas por um valor inteiro que utiliza a representação binária de 5 bits. A quadrícula central da matriz é representada por:

Matriz[1][1] =

1 1 1 1 1 = 31



Se os bits tiverem valores iguais a zero (0), representam a ausência do respectivo item.

Calcular o número de cruzamentos, número de dobras, sobreposições (arestas, arestas-vertices e vértices) e a soma total das arestas;

}

***Algoritmo* de Compactação**

***Entrada**:* graph *G* (genoma g)
***Entrada:** grafo G compactado*

//a origem do sistema cartesiano, no caso em estudo, fica abaixo, na esquerda e as coordenadas estão somente no primeiro quadrante.

Ordena os valores das coordenadas *x* e *y* do grafo *G*;
Os vértices que compartilham as coordenadas *x* mais a direita tem suas coordenadas *x* decrescidas de uma unidade, em um loop, até que esse decréscimo não melhore mais o grafo.
Os vértices que compartilham as coordenadas y mais acima tem suas coordenadas *y* decrescidas de uma unidade, em um loop, até que esse decréscimo não melhore mais o grafo.
Os vértices que compartilham as coordenadas *x* mais a esquerda tem suas coordenadas *x* acrescidas de uma unidade em um loop até que esse acréscimo não melhore mais o grafo.
Os vértices que compartilham as coordenadas *y* mais abaixo tem suas coordenadas *y* acrescidas de uma unidade, em um loop até que esse acréscimo não melhore mais o grafo.
Este procedimento é feito de modo a compactar o grafo da direita para a esquerda, da esquerda para direita, de cima para baixo e de baixo para cima.

***Algoritmo* de Melhoria_local**

Entrada: graph *G* (genoma g)
Entrada: *grafo G melhorado*

Enquanto (graph G ainda obtiver qualquer melhoria no processo abaixo)
{
    Para (cada vértice $v \in V$, do grafo *G*)
    {
        Provocar uma perturbação de uma unidade nas coordenadas dos vértices, em todas as direções, mantendo sempre aquela que provocar melhorias no grafo.
    }
}

***Algoritmo* Loop_reajuste**

Entrada: melhor individuo após o AG convergir
Saida: individuo reajustado

Para (cada aresta $e \in E$ do individuo)
{
    Mutação_arestas;
    Matriz_checagem(Ny);
    Se (individuo melhorou**)**
        Melhor = true;
    else
    {
        Melhor = false;
        contMF++; // É utilizado para contar quantas vezes o individuo melhorou. É utilizado para estabelecer um critério de parada nessa busca.
    }
    Se (**Melhor**)
        O Loop continua, agora, considerando as arestas do novo individuo;
    Se (**contMF** = 10)
        Finalizar processo;
}