Universidade Federal de Goiás
Instituto de Informática

Joelma de Moura Ferreira

Uma Investigação em Otimização Interativa Multiusuário para Desenho de Grafos

Goiânia
2006

Joelma de Moura Ferreira

Uma Investigação em Otimização Interativa Multiusuário para Desenho de Grafos

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Mestrado em Ciência da Computação.

Área de concetração: Ciência da Computação

Orientador: Prof. Hugo Alexandre D. do Nascimento

Co–Orientador: Prof. Eduardo S. de Albuquerque

Goiânia
2006

Joelma de Moura Ferreira

Uma Investigação em Otimização Interativa Multiusuário para Desenho de Grafos

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Mestrado em Ciência da Computação, aprovada em 02 de Junho de 2006, pela Banca Examinadora constituída pelos professores:

Prof. Hugo Alexandre D. do Nascimento
Instituto de Informática – UFG
Orientador

Prof. Eduardo S. de Albuquerque
Instituto de Informática – UFG

Prof. Judith Kelner
Universidade Federal de Pernambuco – UFPE

Prof. Humberto Longo
Universidade Federal de Goiás – UFG

Joelma de Moura Ferreira

Graduou–se em Ciência da Computação na UCG - Universidade Católica de Goiás. Possui especialização em Redes de Computadores pela Universo - Universidade Salgado de Oliveira. Atualmente é professora do curso de Sistemas de Informação das Faculdades Alves Faria.

1. Otimização 2. Otimização interativa 3. Desenho de Grafos 4. Trabalho Cooperativo

Ao nosso Criador, que torna tudo isso possível.

Agradecimentos

Agradeço primeiramente a Deus que é minha Vida, meu Caminho e meu Guia. À minha família, que apesar de não entender muito bem todo esse tempo dedicado aos livros, soube suportar a minha ausência e se calar quando eu precisava de silêncio e sorrir quando precisava de um carinho.

Agradeço ao meu orientador Hugo Alexandre por me ter prestado, durante todo este tempo, toda atenção, sempre de forma solícita e com uma prontidão exemplar.

Aos amigos do mestrado, certamente sentirei falta de todos vocês. Principalmente, Luciano, Karla, Alaor, Raquel, Anibal, Junio e Daniel, pois começamos essa caminhada juntos.

À equipe do Instituto de Informática, funcionários e professores. Lorena, obrigado pelas conversas e conselhos nos dias de emoção e raiva.

Aos meus alunos das Faculdades Alves Faria, que contribuíram para este trabalho através do seu esforço voluntário, durante vários fins de semana e, que no reconhecimento de meu trabalho souberam suportar os momentos de fadiga.

Agradeço também as Faculdades Alves Faria por ter permitido a utilização de seus laboratórios. Afirmo que sem esse apoio teria sido muito difícil concretizar este trabalho.

Ao Ademar Fraga pelas revisões e contribuições neste material.

Agradeço ao professor Eduardo Simões que no momento final veio contribuir com sua serenidade e objetividade. Isso me ajudou a aumentar a confiança de que tudo daria certo.

E finalmente, a professora Ana Paula, pela qual tenho uma grande admiração e que foi o começo de tudo isso. Eu realmente não me esqueço das pessoas que me fazem bem.

Muitas pessoas me ajudaram, agradeço a todos.

Obrigado, eu realmente não sei se este é o fim ou apenas o começo de uma grande jornada, a única certeza é que vocês ficarão em meu coração.

"Mas os mansos herdarão a terra, e se deleitarão na abundância de paz."

Salmos 37:11,
Bíblia Sagrada.

Resumo

Ferreira, Joelma de Moura. **Uma Investigação em Otimização Interativa Multiusuário para Desenho de Grafos**. Goiânia, 2006. 114p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Vários problemas de otimização dependem da ajuda humana para serem efetivamente resolvidos. Para esses problemas, a intervenção humana é necessária a fim de inserir novas restrições e/ou ajudar um método de otimização a produzir resultados de melhor qualidade. Em alguns casos, dois ou mais usuários podem simultaneamente interagir com um sistema para resolver um problema em comum. O presente trabalho investiga a resolução cooperativa (multiusuário) de um problema de desenho de grafos. São estudadas formas de permitir cooperação entre usuários e recursos que facilitem a interação homem-computador e homem-homem neste tipo de atividade. Esta dissertação apresenta ainda uma abordagem geral interativa multiusuário para resolução cooperativa de problemas de otimização, que chamamos Co-*UserHints*.

Palavras–chave

Otimização interativa, Desenho de Grafos, Trabalho Cooperativo Suportado por Computador

Abstract

Ferreira, Joelma de Moura. **An Investigation in Multi-user Interactive Optimization for Graph Drawing**. Goiânia, 2006. 114p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Several optimization problems cannot be fully solved without human aid. For these problems, human intervention is necessary in order to incorporate new domain-knowledge constraints and/or to help an optimization method to produce better solutions. In some cases, two or more users may simultaneously interact with a system for solving a common optimization problem. This thesis investigates the resolution of a graph drawing problem by means of a cooperative (multi-user) approach. It proposes ways of allowing cooperation between users as well as resources for human-computer and human-human interaction in this kind of activity. The thesis also introduces a new general framework for multiuser interactive optimization, that we call Co-*UserHints*.

Keywords

Interactive Optimization, Graph Drawing, Computer Supported Cooperative Work

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO 1

Introdução

A otimização combinatória trabalha com modelos matemáticos que permitem encontrar, em um conjunto finito de soluções possíveis de um problema, aquela que pode ser classificada como a melhor solução. As escolhas são discretas, sendo necessário obedecer algumas restrições.

A junção da ciência da computação com a otimização combinatória produziu bons métodos automáticos de otimização. Entretanto, existem casos para os quais esses métodos são ineficientes, não conseguindo encontrar a melhor solução em um tempo satisfatório.

Para problemas de otimização com vários objetivos e restrições e que estão sujeitos a condições dinâmicas, em geral, os métodos automáticos propostos até o momento não funcionam adequadamente.

Aumentar o poder computacional pode não resolver o problema e até ser impraticável financeira ou tecnologicamente. Uma alternativa seria utilizar a inteligência humana para auxiliar o algoritmo a convergir para a melhor solução. Envolvendo o elemento humano no processo de otimização, é possível ajudar um método semi-automático a escapar de mínimos locais e produzir soluções de alta qualidade. Pesquisas neste sentido têm sido feitas em uma nova área da otimização: a otimização interativa [Anderson et al. 2000, Nascimento 2003].

Existem vários estudos na área de otimização interativa, alguns deles propõem novas abordagens gerais para resolução de problemas de otimização, como o HuGS [Anderson et al. 2000] e *User Hints* [Nascimento 2003], enquanto outros investigam técnicas de interação homem-computador para problemas específicos [Bayazit, Song e Amato 2000]. O Capítulo 3 descreve algumas dessas pesquisas.

Desenho de grafos[1] é um exemplo típico de atividade que envolve diversos problemas de otimização que podem ser melhor resolvidos de forma interativa. A área de desenho de grafos se preocupa em construir representações visuais de

[1]Grafo é um modelo matemático utilizado para representar relações entre objetos concretos e abstratos. Uma introdução sobre desenho de grafos é dada na Seção 2.2.2

grafos, satisfazendo um conjunto de critérios estéticos pré-definidos. Anderson *et.al.* [Anderson et al. 2000] e do Nascimento [Nascimento 2003] propuseram sistemas interativos que combinam as habilidades de um operador humano com recursos de métodos de otimização semi-automáticos.

Os resultados desses sistemas mostram que o elemento humano interagindo com métodos de otimização conseguem, na maioria das vezes, produzir um desenho de melhor qualidade em um tempo inferior, quando comparado ao tempo dos algoritmos executando sozinhos. Contudo, nenhuma das pesquisas considerou a possibilidade de interação envolvendo duas ou mais pessoas. Na verdade, todas as abordagens para otimização interativa conhecidas consideram somente a interação homem-computador mono-usuário.

Estudos na área de CSCW[2] revelam, em contrapartida, que algumas atividades complexas que exigem o conhecimento de diversas áreas, quando abordadas de forma cooperativa, simultaneamente por mais de um usuário em um mesmo ambiente físico, produzem resultados melhores. Isso se deve em parte à possibilidade de interação face-a-face, com utilização de gestos e expressões faciais. Além desses fatores, o compartilhamento de conhecimento e a diversidade de habilidades auxiliam o grupo a alcançar uma mesma meta [Inkpen et al. 1999, Scott, Mandryk e Inkpen 2003].

Como pessoas diferentes podem ter conhecimentos e habilidades diversas a respeito de como desenhar um mesmo grafo, a interação de um grupo auxiliado por computador nesta atividade pode trazer benefícios em relação ao tempo e a qualidade do desenho de grafo produzido. Desta forma, o foco central do presente trabalho é estudar alternativas tecnológicas que permitam a mais de um usuário interagir simultaneamente com métodos automáticos de desenho de grafos.

1.1 Objetivos

Fazem parte dos objetivos desta dissertação:

- Identificar se ocorre algum ganho em se inserir mais de um usuário no processo de otimização de um desenho de grafo, ou seja, verificar se a proximidade dos usuários pode ampliar a interação auxiliando a descoberta da melhor solução e/ou diminuindo o tempo de resolução;

[2] A sigla CSCW significa *Computer Supported Cooperative Work*. Esse campo da ciência estuda os fatores humanos e computacionais envolvidos com o trabalho em grupo auxiliado por computador. A Seção 4.1 trata esse assunto em detalhes.

- Estudar maneiras alternativas automáticas que proporcionem a cooperação entre as pessoas envolvidas no processo de desenho de grafos em um ambiente cooperativo de otimização interativa;
- Propor uma abordagem interativa cooperativa para problemas de otimização que estende a abordagem para otimização interativa *User Hints*, descrita em detalhes na Seção 3.2.1. A nova abordagem será denominada *User Hints* cooperativa (*Co-UserHints*). Ela será utilizada para investigar o problema de desenho de grafos sob o enfoque da otimização interativa multiusuário.

1.2 Organização da Dissertação

O restante deste trabalho está organizado em oito capítulos, como descrito a seguir. O Capítulo 2 contém uma visão introdutória sobre visualização e desenho de grafos. O Capítulo 3 apresenta os conceitos de otimização combinatória e interativa. No Capítulo 4 são abordados tópicos a respeito de trabalho colaborativo. O Capítulo 5 descreve a nova abordagem Co-*UserHints*, a qual permite interação simultânea entre vários usuários e métodos de otimização na resolução de um problema de otimização. O Capítulo 6 apresenta duas aplicações da abordagem *Co-UserHints* para problemas de desenho de grafos direcionados. Dois sistemas chamados *cCo-GDHints* e *iCo-GDHints* são apresetados. A descrição dos testes dos sistemas e os principais resultados da pesquisa são sumarizados no Capítulo 7. No Capítulo 8 são apresentadas as conclusões e trabalhos futuros para esta dissertação.

CAPÍTULO 2

Visualização de Grafos

Neste capítulo são apresentados os fundamentos de três linhas de pesquisa relacionadas ao presente trabalho. Tópicos em visualização de informação, teoria de grafos e desenho de grafos são abordados.

2.1 Visualização de Informação

Milhões de terabytes de dados são gerados globalmente a cada ano e 99% deles é digital [Keim 2002]. Essa grande quantidade de dados torna difícil a sua compreensão, navegação e manipulação. É necessário, criar formas mais eficientes de representá-los que permitam avaliá-los de maneira fácil e rápida.

A forma mais natural é através da representação visual. A ciência "visualização" especifica técnicas que, com o auxílio dos computadores, permitem as pessoas analisarem os dados utilizando imagens [Huang 1999]. A visualização tem como ponto fundamental o uso da capacidade humana de reconhecer rapidamente padrões visuais. Ela investiga técnicas que transformam dados, informação e até conhecimento em um formato visual diminuindo assim a sobrecarga cognitiva do usuário [Gershon e Page 2001].

Card e outros [Card, Mackinlay e Shneiderman 1999] definem visualização como "o uso da representação visual interativa de dados utilizando o computador para aumentar a cognição". A visualização é dividida em duas áreas: visualização científica e visualização de informação. A diferença básica entre as duas está na natureza dos dados de entrada. A primeira trabalha formas de visualizar dados físicos (científicos) e a segunda formas de visualizar dados abstratos (informações). Neste trabalho será focalizado a visualização de informação e em especial duas de suas sub-áreas: a visualização de grafos e desenho de grafos.

A visualização de grafos mostra de forma visual a relação entre entidades de informação, ou seja, trata da representação de estruturas de dados relacionais onde os grafos são a forma fundamental de representação dessas estruturas.

Pela definição dada acima parece que visualização de grafos e desenho de grafos são a mesma coisa, mas existe uma pequena diferença. Herman e outros [Herman, Melançon e Marshall 2000] assinalam esta diferença quando dizem que desenho de grafos se preocupa principalmente em calcular as posições dos vértices e arestas dentro da área 2D ou 3D de visualização do desenho, levando em consideração critérios estéticos, ou seja, se preocupa em investigar métodos automáticos que criem representações visuais de um grafo (ver Seção 2.2.2). Já a visualização de grafos tem o objetivo definir técnicas que ampliem a cognição facilitando a navegação e a interação do usuário com o desenho do grafo em questão.

As duas áreas trabalham em conjunto. A visualização de grafos, por exemplo, é fortemente influenciada pela quantidade de vértices e arestas de um grafo. Um algoritmo pode construir um bom desenho de grafo sem garantir que esse desenho terá uma dimensão utilizável. Assim é necessário usar algumas técnicas de visualização que permitam ao usuário interagir com este desenho [Herman, Melançon e Marshall 2000], entre elas, *Zoom*, *Foco+Contexto*, *Clustering*.

Zoom é uma técnica já tradicional na visualização de grandes estruturas, onde a visualização do todo é substituída pela visualização da parte ampliada. Pode ser classificado em *Zoom* Geométrico que simplesmente aumenta ou diminui a visualização do grafo ou *Zoom* Semântico que modifica o conteúdo da informação mostrando detalhes do grafo [Card, Mackinlay e Shneiderman 1999].

Foco+Contexto também amplia a visualização de parte da estrutura, mas diferente do *Zoom*, não permite a perca do contexto. Por exemplo, uma região de um grafo pode ser ampliada enquanto as outras que não receberam o foco são compactadas [Card, Mackinlay e Shneiderman 1999].

Na técnica de *Clustering* são descobertos grupos de dados baseado em escolhas semânticas e então estes elementos são visualizados como um único vértice. Isso reduz a quantidade de informação visível no grafo, diminuindo a sua complexidade. Para mais detalhes sobre as técnicas de visualização consultar [Card, Mackinlay e Shneiderman 1999, Ware 2004, Herman, Melançon e Marshall 2000].

Existem várias aplicações de grafos na computação como: construção de circuitos VLSI, engenharia de software, modelagem de base de dados, sistemas orientados a objetos, estruturas de dados, sistemas de tempo real, diagrama de fluxo de dados, redes neurais, redes *Bayesianas*, gerenciamento de projeto, lógica de programação, realidade virtual, gerenciamento de documentos entre outros. Tendo também grande importância na biologia e química para a representação de seqüenciamento genético, moléculas, árvores genéticas e tantas outras [Battista et al. 1999, Novák 2002, Herman, Melançon e Marshall 2000].

A Seção 2.2 apresenta mais detalhes sobre os conceitos ligados a grafos, desenho de grafos e os seus principais algoritmos.

2.2 Introdução a Desenho de Grafos

2.2.1 Conceitos de Grafos

Grafos são modelos matemáticos que permitem relacionar entidades de dados. As entidades são identificados como vértices e a relação entre elas como arestas. Abaixo está uma definição de grafo dado por West [West 1996]:

> *Definição*: Um grafo G com n vértices e m arestas consiste de um conjunto de vértices $V(G) = \{v_1, \ldots, v_n\}$ e um conjunto de arestas $E(G) = \{e_1, \ldots, e_m\}$ onde cada aresta possui dois vértices chamados seus extremos. Será denotado de uv para uma aresta $e = \{u, v\}$. Se $uv \in E(G)$, então u e v são adjacentes.

Os grafos podem ser classificados pelo seu tamanho, estrutura e tipo de arestas. De acordo com seu tamanho os grafos podem ser [Marshall 2001]:

- Pequeno - grafos até 100 elementos (vértices e arestas);
- Médio - grafos com até 1000 elementos (vértices e arestas);
- Grande - grafos com até 10.000 elementos (vértices e arestas);
- Esparso - grafos com mais de 10.000 elementos (vértices e arestas).

Os conceitos a seguir foram retiradas de [West 1996] e têm relação com a topologia de um grafo, sendo importantes no restante deste texto.

Um grafo é dito *finito* se o seu conjunto de arestas e vértices é finito. Um *loop* é uma aresta cujos vértices extremos são iguais. Seja uv uma aresta de um grafo G, essa aresta é dita *adjacente* a u e adjacente a v, também se diz que uv é *incidente* em u e v.

O grau de um vértice v, escrito $d(v)$, é o número de arestas (somente aquelas que não são *loops*) adjacentes a v. *Arestas paralelas* são arestas que têm o mesmo par de vértices adjacentes. Um grafo é dito *simples* se não possui *loops* nem arestas paralelas.

Grafo direcionado é aquele onde cada aresta é um par ordenado de vértices, ou seja, cada aresta possui uma direção. A aresta $uv \in E(G)$ de um grafo direcionado G sai do vértice u e vai para o vértice v. O vértice u é chamado origem e v destino.

A escolha da origem e do destino determinam a direção da aresta, assim $uv \in E(G)$ é diferente da aresta $vu \in E(G)$.

Um *subgrafo* de um grafo G é um grafo H tal que $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Um *grafo completo* é um grafo simples em que todo par de vértices possui uma aresta entre eles. Um grafo é dito *planar* se for possível desenhá-lo no plano sem cruzamentos de arestas.

Um *caminho* de tamanho k é uma seqüencia de vértices e arestas distintas v_0, e_1, v_1, e_2,... , e_k, v_k tal que $e_i = v_{i-1}v_i$ para todo $1 \leq i \leq k$. Um caminho denominado $u, v - path$ tem o primeiro vértice em u e o último em v. Um caminho é *fechado* se tem tamanho no mínimo um, e seus vértices extremos são iguais. Um *ciclo* é um caminho fechado em que os únicos vértices que se repetem são o primeiro e o último do caminho. Um grafo que não contém ciclos é chamado *acíclico*.

A *distância* entre dois vértices em um grafo, também chamado distância teórica, é o tamanho do caminho mais curto entre estes vértices.

Um grafo G é dito *conexo* se ele tem um $u, v - path$ para cada par de vértices u, $v \in V(G)$. Um grafo conexo acíclico é chamado *árvore*.

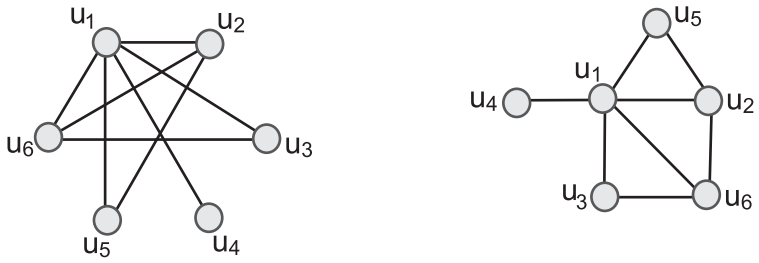
Esta pesquisa trabalha com grafos finitos simples conexos de tamanho pequeno, que serão identificados a partir de agora somente por grafo, caso diferenciados serão explicitamente informados no texto.

2.2.2 Desenho de Grafos

Desenho de grafos é a representação física de um grafo, sendo a principal forma visual de se mostrar a relação entre entidades. Um desenho de grafo é formado através da atribuição de posições no espaço (bi ou tri-dimensional) a essas entidades [Herman, Melançon e Marshall 2000]. As entidades são representadas no grafo por vértices e arestas.

Um grafo pode ser representado visualmente de várias formas diferentes e a utilidade de um desenho de grafo depende da capacidade que se tem de rapidamente e claramente perceber o seu significado [Webber 1998]. Na Figura 2.1 vemos dois desenhos diferentes de um mesmo grafo.

Comparando a idade dos estudos sobre teoria e algoritmos de grafos com o problema de desenho de grafos, esta última é considerada uma área relativamente jovem, mas muito dinâmica e com resultados expressivos. Ela tem ampla aplicação em vários campos da ciência, indo desde a ciência social até a área de engenharia, passando inclusive pela ciência biológica.



$G = (V,E)$

$V = \{u_1, u_2, u_3, u_4, u_5, u_6\}$

$E = \{(u_1,u_2), (u_1,u_3), (u_1,u_4), (u_1,u_5), (u_1,u_6), (u_2,u_5), (u_2,u_6), (u_3,u_6)\}$

Figura 2.1: *Desenhos diferentes de um mesmo grafo.*

2.2.3 Critérios Estéticos

Para gerar um desenho de grafo leva-se em consideração principalmente as características estéticas e restrições com relação a posicionamento de vértices. Essas características e restrições, muitas vezes, são definidas com base na aplicação a que o desenho se destina, pois a aparência do desenho pode influenciar o entendimento das informações modeladas. Assim, qual é a melhor representação visual para um grafo? Não existe uma resposta: depende da finalidade do desenho [Battista et al. 1999].

Os critérios estéticos são um conjunto de propriedades gráficas de um desenho de grafo que podem tornar-lo mais agradável esteticamente para a maioria das aplicações. Battista *et. al.* [Battista et al. 1999] enumeram os principais critérios estéticos que definem um bom desenho de grafo:

- ausência ou redução de cruzamentos entre arestas;
- tamanho de arestas uniformes;
- ausência ou redução de curvas feitas por arestas;
- maximização do ângulo entre arestas;
- minimização da soma do tamanho das arestas;
- minimização do tamanho máximo de uma aresta;
- uniformidade na direção das arestas (para grafos direcionados);
- simetria do desenho e;
- redução da área total do desenho.

Muitos desses critérios são conflitantes entre si. Por exemplo, manter a uniformidade na direção das arestas em um grafo direcionado pode significar aumentar a quantidade de cruzamentos [Battista et al. 1999].

Purchase *et. al.* [Purchase, Cohen e James 1995] *apud* [Webber 1998] conduziram alguns experimentos que tinham como objetivo identificar os benefícios de alguns critérios na compreensão do desenho. Eles organizaram experimentos onde usuários precisavam resolver um problema de teoria de grafos. Cada usuário trabalhava em desenhos que variavam em um dos critérios tendo os outros fixos. Através dos resultados, Purchase *et. al.* definiram uma ordem de importância de alguns critérios. Segundo eles a ordem seria: minimização de cruzamentos, minimização de curvas e maximização de simetria.

Contudo, apesar deste guia de prioridade de critérios proposto, é importante destacar o fato de um bom desenho de grafo depender realmente da aplicação para a qual ele é criado.

2.2.4 Estilos de um Desenho de Grafos

Além dos critérios estéticos, um desenho de grafo pode ainda ser classificado conforme um estilo, sendo este, um dos principais pontos que determiná a seleção do algoritmo a ser usado para desenhar um grafo.

- **Linha reta**: cada aresta do desenho é um único segmento de linha reta;
- **Poligonal**: cada aresta é mapeada para um ou mais segmentos de reta;
- **Curva**: cada aresta é representada por uma curva;
- **Ortogonal**: é um desenho de grafo linha reta e poligonal onde cada segmento de reta é paralelo a um dos eixos do espaço de visualização do desenho. O encontro de dois segmentos de retas de sentidos diferentes de uma mesma aresta é chamado curva;
- **Grade**: é um desenho onde cada vértice e curva de arestas estão em uma coordenada inteira do espaço de visualização do desenho;
- **Visibilidade**: os vértices são representados por barras horizontais, se existir uma aresta entre dois vértices existirá uma linha vertical que ligará as duas barras horizontais, sem tocar qualquer outra;
- **Superfície**: os grafos são representados por superfícies 3D, como esferas, cones, cilindros, poliedros e toris.

Esses estilos podem também ser combinados, e a Figura 2.2 mostra um mesmo grafo desenhado em estilos diferentes.

O direcionamento vertical das arestas é um estilo importante associado a desenhos de grafos direcionados. Um desenho direcionado é dito *upward* (ou *downward*) se as suas arestas são monotonicamente crescentes (ou descrescentes).

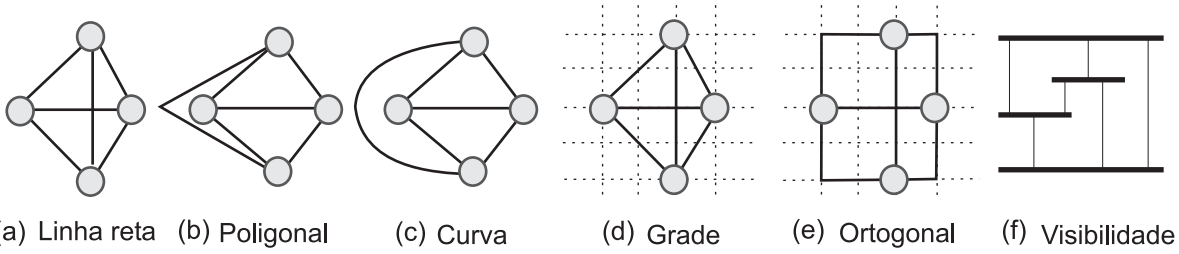


Figura 2.2: *Estilos de um desenho de grafo.*

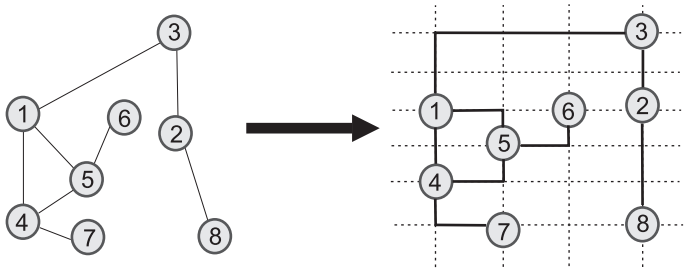


Figura 2.3: *Ortogonalização de um grafo geral.*

2.2.5 Paradigmas para Desenho de Grafos

Um algoritmo para desenho de grafo leva em consideração os estilos apresentados na seção anterior. Porém, ele também pode ser classificado de acordo com alguns paradigmas que serão usados para montar o desenho. Cada paradigma objetiva satisfazer um ou mais critérios estéticos. Existem vários paradigmas para algoritmos de desenho de grafos. Abaixo são listados alguns deles retirados de [Battista et al. 1999, Kaufmann e Wagner 2001]:

Topologia-Forma-Métrica

Este paradigma é utilizado para construir desenhos ortogonais em grade, levando em consideração a topologia, forma e métrica do desenho. Um desenho de grafo geral é transformado em um desenho ortogonal em grade através de três passos: planarização, que descobre a topologia do desenho, transformando-o em planar; ortogonalização que define a forma do desenho e por fim a compactação que determina a métrica final. A Figura 2.3 mostra a transformação de um desenho de grafo geral em ortogonal em grade.

Hierárquico

Neste tipo de abordagem os vértices do desenho são posicionados em níveis hierárquicos horizontais chamados de camadas. A ordenação dos vértices dentro de cada camada deve ser feita de tal forma que minimize os cruzamentos de

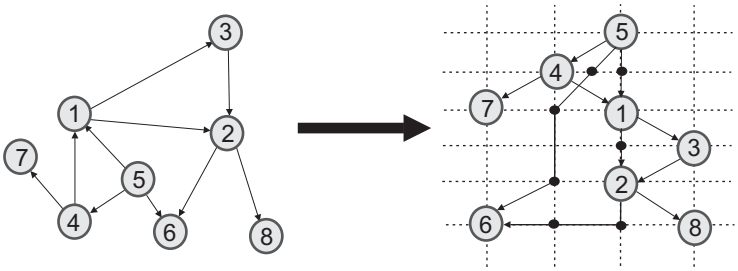


Figura 2.4: *Hierarquização de um grafo direcionado.*

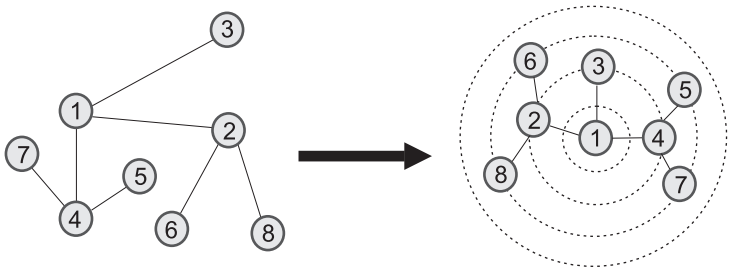


Figura 2.5: *A transformação do desenho de uma árvore em um desenho radial.*

arestas. Durante o processo de hierarquização (*Layering*) de um desenho, quando uma aresta passa por várias camadas, pode existir a necessidade de inserção dos chamados *vértices falsos* na junção da aresta com essas camadas. Esse procedimento é necessário para alguns algoritmos que trabalham com desenhos hierárquicos, como o algoritmo Sugiyama (descrito em [Battista et al. 1999]). Os algoritmos construídos segundo esta metodologia possuem os seguintes passos: remoção de ciclos, associação dos vértices em camadas, redução de cruzamentos e definição da coordenada x dos vértices. A Figura 2.4 mostra a hierarquização de um desenho de grafo direcionado. Os círculos cheios na junção das arestas com as camadas representam os vértices falsos. Este modelo é bastante usado para grafos direcionados.

Radial

O desenho radial é mais utilizado em árvores. Também utiliza camadas, mas as camadas são agora círculos concêntricos em que o vértice raiz é colocado na origem dos círculos e os seus filhos, na camada imediatamente seguinte, isso ocorre com todos os vértices, a partir de então. A Figura 2.5 mostra o desenho de uma árvore no formato radial.

HV-*drawing*

Este paradigma é utilizado para desenhar árvores binárias. Os vértices filhos de cada vértice são alinhados horizontalmente (à direita) e verticalmente (abaixo), por isso o nome HV. Os desenhos formados através desse paradigma são planares,

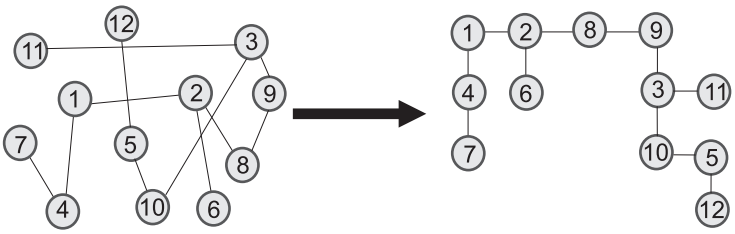


Figura 2.6: *Desenho HV de uma árvore binária.*

linha reta, ortogonais e *downward* (em caso de grafo direcionado), como visto na Figura 2.6.

Visibilidade

Este paradigma cria um desenho poligonal. O primeiro passo, depois de gerada uma solução inicial, é a planarização desse desenho. O segundo passo, chamado visibilidade, transforma cada vértice em uma barra horizontal, inclusive os vértices falsos, e as arestas em barras verticais que conectam as horizontais. O último passo cria o desenho final, transformando as barras novamente em vértices e arestas.

Augmentation

Tem como primeiro passo a planarização do desenho de grafo. O segundo passo, que dá o nome ao paradigma (*Augmentation*), introduz arestas e vértices falsos, para que toda as faces do desenho tenham três lados. No terceiro e último passo chamado, triangularização, o desenho final é produzido usando propriedades geométricas dos triângulos formados. Os vértices e arestas falsos são removidos.

Divisão e Conquista

Na divisão e conquista o grafo é dividido em subgrafos menores que são então redesenhados. Depois os subgrafos são recombinados, formando o desenho final. Este paradigma é muito utlizado para desenhar árvores e existem diversas técnicas que privilegiam vários critérios estéticos.

Força-Direcionada

Neste paradigma o desenho de grafo é modelado como um sistema de forças físicas. Os algoritmos construídos seguindo esta abordagem são identificados como algoritmos de forças direcinado e trabalham a partir de um desenho aleatório, utilizando o sistema de força modelado, movem os vértices desse desenho para outras posições até que o sistema alcance o equilíbrio. Um dos algoritmos de otimização

utilizados nesta pesquisa pertence a classe dos métodos de força direcionada. Os algoritmos de força direcionada são descritos em detalhes na Seção 2.2.7.

2.2.6 Algoritmos para Desenho de Grafos

A área de pesquisa em desenho de grafos é muito dinâmica e existem inúmeros algoritmos que levam em consideração um ou mais critérios estéticos. Herman *et. al.* [Herman, Melançon e Marshall 2000] afirmam que um algoritmo para desenho de grafos deve levar em consideração três questões: preservação do mapa mental, complexidade e restrições.

Segundo Webber [Webber 1998] "o mapa mental do usuário é um modelo semântico do espaço de informação que ele desenvolve para interpretar um desenho de grafo". Isso significa que o usuário possui uma percepcão particular do desenho e a utiliza para interpretar as relações existentes neste. O mapa mental é como uma "fotografia" que o usuário possui agregada ao seu significado. Desta forma, duas execuções de um algoritmo em um mesmo desenho não devem mudar radicalmente a representação visual desse grafo [Herman, Melançon e Marshall 2000]. Algoritmos que perturbam de maneira drástica um desenho de grafo podem destruir o mapa mental que o usuário tem do desenho, prejudicando a sua compreensão.

Além do algoritmo preservar o mapa mental do usuário, ele deve, também, executar em um tempo polinomial[1]. Isso é necessário pois algoritmos que levam muito tempo para finalizar o processamento de uma tarefa, quando aplicados a problemas com grande volume de dados, podem prejudicar o desempenho do processo de otimização como um todo.

Contudo, os algoritmos para desenho de grafos que possuem muitas restrições e critérios estéticos são complexos de serem implementados na prática, obter o melhor desenho em alguns casos é NP-completo ou NP-difícil [Battista et al. 1999]. Por exemplo, segundo Battista *et. al* [Battista et al. 1999] minimizar o número de cruzamentos é NP-completo se o grafo for hierárquico com somente duas camadas; minimizar a área de um desenho em grade ou minimizar o tamanho máximo de uma aresta são problemas NP-difíceis.

Existem várias heurísticas para otimizar um desenho de grafo, heurísticas essas que são constantemente melhoradas pela comunidade científica. A metodologia de força direcionada é uma das principais técnicas para construção de algoritmos para desenho de grafos. Essa técnica será discutida em mais detalhes na próxima seção, onde também serão definidas algumas de suas heurísticas.

[1]Para mais informações sobre algorimos polinomiais consultar [Garey e Johnson 1979]

2.2.7 Forças Direcionadas

Os algoritmos baseados em forças são muito utilizados devido a sua simplicidade e facilidade de implementação. São métodos usados principalmente para desenhar grafos em linha reta de tamanho pequeno (ver nas Seções 2.2.1 e 2.2.4), produzindo bons resultados na maioria das vezes. Modelam o desenho como um sistema de forças físicas, em que as arestas e vértices são objetos que possuem forças atuando sobre eles. Os algoritmos construídos dentro desta abordagem procuram levar o sistema de forças a um equilíbrio e, dessa forma, conseguir um desenho com maior simetria e livre de cruzamentos [Battista et al. 1999].

Muito da simplicidade dos algoritmos de forças direcionadas está ligada à analogia que fazem com as forças físicas, o que os tornam intuitivos e fáceis de programar. Esse tipo de abordagem possui dois componentes: um modelo de forças e uma técnica para alcançar a convergência para o melhor desenho (equilíbrio). Desta forma, os métodos de força direcionada se diferenciam pelo modelo que implementam e pela forma como alcançam o equilíbrio. Por isso, produzem desenhos diferentes, privilegiando critérios distintos [Aiello e Silveira 2004]. Entretanto, no geral, esses algoritmos costumam produzir desenhos simétricos com tamanhos uniformes das arestas e vértices equiespaçados.

Contudo, essa busca de simetria e uniformidade de arestas pode muitas vezes produzir um desenho de grafo com uma grande quantidade de cruzamentos. Neste caso, algoritmos com modelos de força mais complexos podem ser usados, como métodos genéticos ou *simulated annealing*. Esses algoritmos mais complexos são mais difíceis de serem parametrizados e costumam ser mais lentos na sua execução [Bertault 1999].

Existem vários algoritmos de forças direcionadas. Abaixo é apresentada uma lista dos métodos clássicos. Os estudos sobre este tipo de técnica, geralmente, envolvem a modificação e adaptação desses algoritmos.

- ***Spring Embedder*** [Eades 1984]: primeiro algoritmo de forças, foi proposto por Eades, que modelou o sistema considerando os vértices como partículas elétricas e as arestas como molas. As partículas elétricas sofrem uma força de repulsão entre elas, e ao mesmo tempo, uma força de atração em relação às partículas elétricas adjacentes, proveniente das molas de conexão.
- **Davidson e Harel** [Davidson e Harel 1996]: é um método baseado no *simulated annealing* [Aiello e Silveira 2004] em que os critérios estéticos são modelados por valores independentes, que são depois unidos em uma única função de energia geral;

- **Kamada e Kawai** [Kamada e Kawai 1989]: baseado no *Spring Embedder*, esse método trabalha com um sistema de energia buscando um estado de equilíbrio. Para cada par de vértices adjacentes, o tamanho da mola (aresta) relaxada é proporcional ao caminho mais curto no grafo entre eles. Focaliza a minimização dos cruzamentos, simetria do desenho e uniformização do tamanho das arestas;
- **Frutchterman e Reingold** [Fruchterman e Reingold 1991]: trabalha com um sistema de atração eletrostática em que a força de atração entre os vértices conectados é balanceada pela força de repulsão entre todos os vértices. A determinação do equilíbrio é baseado no *simulated annealing*;
- **Forças Magnéticas**: proposto por Sugiyama e Misue [Sugiyama e Misue 1994], neste modelo algumas arestas encontram-se magnetizadas e um campo magnético atua sobre elas. É utilizado para desenhos de grafos direcionados, pois permite dar sentido as arestas conforme a orientação do campo magnético.

O presente trabalho utiliza uma variação dos algoritmos *Spring Embedder* e Davidson e Harel como métodos de otimização. As subseções seguintes descrevem esses dois métodos clássicos de forças direcionadas, em detalhes. Os algoritmos que serão apresentados foram descritos por Aiello e Silveira [Aiello e Silveira 2004].

A partir destes ponto será utilizando a seguinte notação: $G = (V, E)$ é um grafo sendo V o conjunto dos vértices e E o conjunto das arestas, com $u, v \in V$. A posição no plano (coordenadas (x, y)) de um vértice $u \in V$ em um desenho D de G é representado por $p_u^D = (x_u, y_u)$, no caso do desenho estar subtendido será escrito somente p_u. A notação $\overrightarrow{p_u p_v}$ indica o vetor unitário $\frac{p_v - p_u}{||p_v - p_u||}$, e $d(u, v)$ a distância euclidiana entre as posições do vértice u e v, sendo $d(u, v) = ||p_u - p_v||$.

Spring Embedder

Primeiro método de forças direcionadas, criado por Eades [Eades 1984] em 1984. Modela o grafo como sistema de forças composto por molas e partículas elétricas sobre as quais atuam forças de atração e repulsão. As molas representam as arestas que agem em seus vértices adjacentes com uma força logarítmica de atração. Cada vértice representa uma partícula elétrica, os vértices não adjacentes repelem-se com o quadrado do inverso da força, é como se existisse uma mola de tamanho infinito ligando os vértices não adjacentes.

O algoritmo de Eades cria aleatoriamente uma solução inicial. Um número fixo de iterações é executado e a cada passo todos os vértices são movidos simultâneamente na proporção da força resultante exercida neles. A força resultante é a soma de todas as forças de repulsão e atração no vértice. O objetivo é encontrar

um desenho onde as posições dos vértices produzam a força total do sistema, sendo zero ou próximo de zero, ou seja, até uma configuração que estabeleça o equilíbrio do sistema [Aiello e Silveira 2004].

Os parâmetros que controlam as forças que agem nas partículas causando o seu movimento são: o tamanho da mola, a firmeza (tipo) da mola, o tamanho da repulsão elétrica entre os vértices e a configuração inicial do sistema.

Os vértices adjacentes possuem uma força de atração $f_a(u, v)$ segundo a fórmula:

$$f_a(u, v) = c_1 \log\left(\frac{d(u, v)}{c_2}\right) \overrightarrow{p_v p_u} \tag{2-1}$$

Sendo c_1 a constante que controla a firmeza da mola e c_2 a constante que representa o tamanho natural da mola.

Todo par de vértices não adjacente possui uma força de repulsão $f_r(u, v)$ segundo a fórmula:

$$f_r(u, v) = \left(\frac{c_3}{d^2(u, v)}\right) \overrightarrow{p_u p_v} \tag{2-2}$$

Sendo c_3 uma constante que controla o tamanho da repulsão.

A força f_u em um vértice u é calculado pela soma das forças de atração e repulsão:

$$f_u = \sum_{(u,v)\in E} f_a(u, v) + \sum_{(u,v)\notin E, u\neq v} f_r(u, v) \tag{2-3}$$

O algoritmo para este método é apresentado a seguir.

2.1: $SpringEmbedder(G, D)$

Entrada: Desenho D de um grafo $G = (V, E)$.

Saída: Desenho de grafo D.

```
1  contador = 1
2  enquanto contador > M faça
3  |  para cada u ∈ V faça
4  |  |  Calcular a força resultante f(u) das molas conectadas ao
   |  |  vértice u no desenho D
5  |  fim
6  |  para cada cada v ∈ V faça
7  |  |  p_u = p_u + cf(u);
8  |  fim
9  |  contador = contador + 1
10 fim
11 retorna D
```

Onde c e M são constantes. Segundo [Eades 1984] esse algoritmo produz bons resultados, em um tempo polinomial, para a maioria dos grafos com em média cinqüenta vértices, contudo, não trabalha muito bem com grafos acima desse valor. O critérios estéticos privilegiados são: uniformizar o tamanho das arestas, minimizar a área e mostrar a simetria do desenho [Aiello e Silveira 2004].

Davidson e Harel

O algoritmo Davidson e Harel [Davidson e Harel 1996] é um método baseado no *simulated annealing* [Aiello e Silveira 2004] em que os critérios estéticos são modelados distintamente por valores independentes, que são depois unidos em uma única função de energia geral. Cada critério estético deve ter um potencial que representa a função de energia.

A partir de um estado inicial, a energia do sistema se altera, sendo que tanto os movimentos que diminuem quanto os que aumentam a energia podem ser aceitos. Os que diminuem são sempre aceitos, os que aumentam são aceitos aleatoriamente considerando a diferença de energia e temperatura atual [Aiello e Silveira 2004].

Cinco critérios estético são avaliados neste método:

- **Disposição uniforme dos vértices:** esse potencial é definido como U_r e calculado para cada par de vértices $(u, v) \in V$. Sua função é evitar que os vértices fiquem muito próximos uns dos outros. Sua fórmula é definida abaixo:

$$U_r(u,v) = \frac{1}{d^2(u,v)} \tag{2-4}$$

- **Espaço disponível:** representado como U_b esse potencial evita que os vértices saiam do espaço de visualização do desenho. Tem o papel de repelir os vértices das bordas da área útil do desenho. A fórmula é descrita a seguir:

$$U_b(u) = \frac{1}{r_u^2} + \frac{1}{l_u^2} + \frac{1}{i_u^2} + \frac{1}{s_u^2} \tag{2-5}$$

  Onde r_u, l_u, i_u, s_u são as distâncias entre a posição do vértice u e a borda direita, esquerda, inferior e superior respectivamente.

- **Penalização de arestas grandes:** esse potencial, representado como U_e, não fixa o tamanho das arestas mas penaliza as arestas muito grandes. É definido para cada par de vértices (u,v) conectados por uma aresta. Sua fórmula é descrita abaixo:

$$U_e(u,v) = d^2(u,v) \tag{2-6}$$

- **Minimização do número de cruzamento de arestas:** representa a quantidade de cruzamentos. Na fórmula abaixo é denotado por U_c.

$$U_c = c \tag{2-7}$$

- **Distância entre arestas:** representado por U_{en} na fórmula abaixo, penaliza as arestas que quase se cruzam, ou seja, que estão praticamente sobrepostas.

$$U_{en}(e,w) = \frac{1}{g_{ew}^2} \tag{2-8}$$

  Onde g_{ew} é a distância mínima entre a posição de w e qualquer ponto sobre a aresta e.

Baseado nos cinco potenciais estéticos listados acima a função de energia é:

$$\begin{aligned} E &= c_c U_c \\ &+ c_r \sum_{u,v \in V, u \neq v} U_r(u,v) \\ &+ c_b \sum_{u \in V} U_b(u) \\ &+ c_e \sum_{(u,v) \in E} U_e(u,v) \end{aligned}$$

$$+ \quad c_{en} \sum_{e=(u,v)\in E, w\in V} U_{en}(e,w). \tag{2-9}$$

Sendo as constantes c_r, c_b, c_e, c_c e c_{en} reguladoras do peso relativo de cada termo dentro da função de energia.

2.2: $DavidsonHarel(G, D)$

Entrada: Desenho D de um grafo $G = (V, E)$.

Saída: Desenho de grafo D.

```
1  Determinar temperatura inicial t = t_0
2  Determinar uma posição randomica a cada vértice de G
3  i = 1
4  enquanto i > M faça
5      j = 1
6      enquanto j > T faça
7          para cada v ∈ V faça
8              p_ant = p_v
9              p_v = p_v + δ_rand
10             se E(p_ant) < E(p_v) então
11                 Com probabilidade 1 − e^((E(p_ant)−E(p_v))/t) , retornar
                   p_v = p_ant
12             fim
13         fim
14         j = j + 1
15     fim
16     Reduzir temperatura t
17     i = i + 1
18 fim
19 retorna D
```

O algoritmo Davidson e Harel é apresentado acima, onde M e t_0 são constantes e $E(p_v)$ é a energia resultante ao mover o vértice v para a posição p. T é a quantidade de iterações com uma mesma temperatura. δ_{rand} é um vetor que determina quanto e em que direção deve-se mover o vértice. Este vetor é calculado aleatoriamente entre todos os vértices que estão em uma circunferência de raio r, sendo que r vai decrescendo em, proporção da temperatura t. A idéia é que, no começo, os movimentos sejam grandes e, à medida que passa o tempo e o desenho alcança uma configuração de energia mínima, e os movimentos sejam cada vez mais curtos.

O algoritmo Davidson e Harel produz bons desenhos topologicamente, mas não esteticamente. Uma outra desvantagem é o alto custo computacional [Aiello e Silveira 2004].

Segundo Aielo [Aiello e Silveira 2004], apesar dos bons resultados que os algoritmos de forças direcionadas produzem, eles contêm algumas desvantagens:

- boa parte dos métodos tem um alto custo computacional;
- como são heurísticas, não existe garantia da otimalidade;
- dificuldade de inserção de novas restrições;
- costumam produzir soluções com cruzamentos, mesmo que o grafo seja planar;
- tratam de forma efetiva somente os tipos de grafos para os quais foram idealizados.

CAPÍTULO 3

Otimização

Neste capítulo são abordados os conceitos sobre otimização combinatória e interativa. Descreve ainda a abordagem *User Hints* como um modelo para otimização interativa.

3.1 Otimização Combinatória

A otimização combinatória tem como objetivo encontrar uma solução "ótima" para um conjunto variáveis de decisão, que otimizam (maximizam ou minimizam) uma função objetivo, sujeitas a algumas restrições.

Segundo Aardal *et. al.* [Aardal et al. 1997], o estudo da otimização combinatória se justifica em função de dois fatores principais: o aumento no poder computacional e ao grande potencial prático das técnicas de otimização combinatória atuais. Esses dois fatores permitem hoje investigar problemas que há dez anos não podiam ser resolvidos, tais como problemas práticos de planejamento espacial, projeção de sistemas de distribuição de energia elétrica, posicionamento de satélites, alocação de recursos, desenvolvimento de circuitos VSLI, empacotamento de caixas em *containers*, corte de vidros e sequenciamento de genes e de DNA, entre outros.

Segundo Bertsekas [Bertsekas 1998], existem muitos algoritmos para problemas de otimização combinatória. Esses algoritmos podem ser classificados em exatos, aproximativos e heurísticos. Os algoritmos exatos garantem encontrar a solução ótima, através da enumeração de todas as soluções possíveis. Entretanto, a maioria dos problemas de otimização de caracter prático são NP-difíceis [Leigh et al. 2002, Garey e Johnson 1979] e, neste caso, a utilização de algoritmos exatos se torna impraticável; dependendo do tamanho da instância de entrada do problema, esses métodos podem necessitar de um tempo consideravelmente elevado para produzir uma solução [Blum e Roli 2003].

Tanto os algoritmos aproximativos quanto as heurísticas destinam-se a encontrar uma boa solução em tempo razoável [Blum e Roli 2003]. Esses métodos, ao contrário dos algoritmos exatos, não vasculham todo os espaço de solução do

problema. A diferença básica entre a técnica aproximativa e a heurística é que as soluções produzidas pelos métodos aproximativos possuem um certo grau de otimalidade; garantia de qualidade essa que não é dada pelas heurísticas [Bertsekas 1998].

Os métodos heurísticos em geral são eficientes e rápidos. Contudo, via de regra, possuem dificuldades de escapar de mínimos locais[1]. São criados para solucionar problemas específicos, geralmente NP-difíceis[2] como: planejamento espacial, projetos de sistemas de distribuição de energia elétrica, posicionamento de satélites, projetos de computadores e de chips VLSI, roteamento ou escalonamento de veículos, alocação de trabalhadores ou máquinas a tarefas, empacotamento de caixas em *containers*, corte de barras e placas, sequenciamento de genes e DNA, classificação de plantas e animais, etc. Essa característica dificulta a sua utilização em outras aplicações [Goldbarg e Luna 2005]. A exceção são as heurísticas gerais, chamadas meta-heurísticas.

As meta-heurísticas são estratégias bem definidas para construção de algoritmos heurísticos e podem ser utilizadas principalmente para aqueles problemas de otimização que não possuem algoritmos eficientes para resolvê-los. Hertz *et. al.* [Hertz e Widmer 2003] definem meta-heurísticas como:

> "Técnicas de otimização combinatória gerais que não estão dedicadas à solução de um problema particular, mas são preferencialmente desenvolvidas com o objetivo de serem flexíveis o suficiente para serem utilizadas nos mais diferentes problemas combinatórios quanto possível."

São exemplos de meta-heurísticas:

- **Busca Construtiva Gulosa:** Constrói uma solução, elemento por elemento A cada passo é adicionado o "melhor" elemento candidato escolhido segundo um certo critério. O método se encerra quando todos os candidatos forem analisados;
- ***Simulated Annealing*:** faz uma analogia entre o processo físico *annealing* de sólidos e a resolução de problemas de otimização combinatorial;
- **Algoritmos Genéticos:** são uma versão computacional do que se passa na natureza. São inspirados na evolução natural dos seres vivos como uma forma de resolver problemas de procura adaptativa de uma solução;

[1] Blum e Roli [Blum e Roli 2003] definem mínimo local como sendo uma solução $s*$ tal que $\forall s \in N(s*) : f(s*) \leq f(s)$, sendo $N(s*)$ a vizinhança de $s*$ e f a função objetivo. Uma vizinhança é um mapeamento $N : S \rightarrow 2^S$ que leva as soluções de S em um subconjunto deste mesmo conjunto de soluções, assim $N(s) = s_1, s_2, \ldots, s_k$ soluções vizinhas de s. A estrutura de vizinhança varia de acordo com o problema tratado

[2] Mais informações sobre problemas NP-difíceis podem ser encontradas em [Garey e Johnson 1979]

- **Busca Tabu:** são metaheurísticas que trabalham com o melhoramento local. Se baseia em iterações proibidas para avançar em direção a solução ótima. Uma iteração consiste em explorar a totalidade da vizinhança da solução para encontrar o vizinho que propõe o melhor melhoramento da função objetivo.

Este trabalho utiliza a meta-heurística de busca gulosa em um algoritmo para integração de desenhos de grafos. O Capítulo 6 discute em mais detalhes esse assunto.

3.2 Otimização Interativa

Apesar do desempenho satisfatório e da grande utilização das meta-heurísticas, os problemas de otimização não são sempre triviais e à medida que a instância de entrada cresce, a dificuldade em se encontrar uma solução de qualidade aumenta em proporção bastante superior, o que demanda uma capacidade de processamento elevada. Além disso, em algumas situações, principalmente aquelas envolvendo sistemas dinâmicos, a modelagem computacional de um problema de otimização prático é complicada e demanda intervenção humana.

Nesse contexto, estudos recentes têm sido feitos no sentido de inserir o homem no processo de otimização, permitindo que este refine um problema (adicionando novas condições dinâmicas) ou guie a convergência de um método semi-automático para uma solução "ótima" [Nascimento e Eades 2003, Scott, Lesh e Klau 2002]. A Figura 3.1 mostra dois modelos de otimização, o tradicional e o interativo.

Nascimento [Nascimento 2003] identificou duas metas fundamentais da otimização interativa:

- Refinar o problema de otimização através da inserção do "conhecimento de domínio" que representa os interesses do usuário;
- Ajudar na convergência para a solução ótima de maneira mais rápida.

Também segundo Nascimento, foi no fim da década de 1980, que os primeiros sistemas interativos de otimização combinatória começaram a aparecer e, apesar de não generalizarem o conceito de otimização interativa, já explicavam a idéia de combinar a inteligência humana com o poder computacional dos métodos tradicionais.

Contudo, foi a partir de 1995 que os conceitos envolvendo interação homem-computador para problemas de otimização começaram a tomar forma no campo científico [Nascimento 2003]. Vários estudos investigaram meios efetivos de permitir

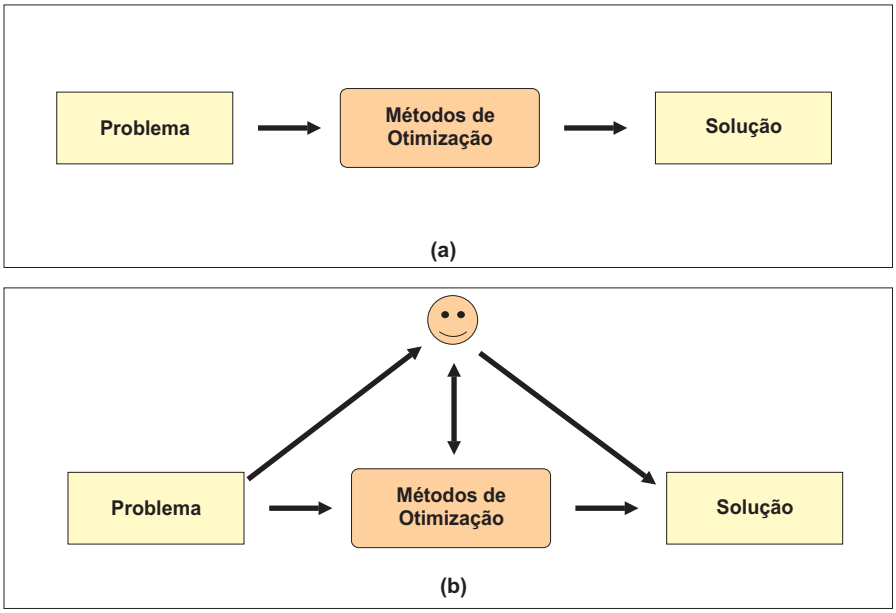


Figura 3.1: *(a) Modelo de otimização tradicional. (b) Modelo de otimização interativa.*

que o homem interaja com métodos automáticos para produzir soluções de alta qualidade para problemas de otimização. Nascimento [Nascimento 2003] descreve algumas das primeiras abordagens interativas para esse fim, entre elas: a abordagem *Interactive Evolutionary*, com aplicações em desenhos de grafos [Jacobsen 2001, Barbosa e Barreto 2001, Rosete-Suarez, Sebag e Ochoa-Rodriguez 1999], o trabalho de Bayazit *et. al.* [Bayazit, Song e Amato 2000] sobre problemas de planejamento de movimentos e a abordagem geral *Human-Guided Search* [Anderson et al. 2000]. Nascimento [Nascimento 2003] apresenta inclusive uma abordagem própria para otimização interativa, chamada *User Hints*.

O trabalho conduzido por Bayazit *et. al.* [Bayazit, Song e Amato 2000], por exemplo, apresenta uma abordagem interativa para um problema de planejamento de movimentos que agrega elementos de interação homem-computador, dispositivos hápticos [Massie e Salisburg 1994] e representação visual. O problema consiste em encontrar um caminho que leva um objeto de um estado inicial (ou localização) S_0 para um estado final S_f, evitando colisões com obstáculos. O problema torna-se complexo quando existe um grande número de configurações a explorar, e quando o caminho a ser construído é dependente de configurações críticas difíceis de serem modeladas. A abordagem desenvolvida pelos pesquisadores explora recursos do sistema visual humano e métodos automáticos, para encontrar um caminho livre de colisão.

Já a abordagem *HuGs (Human Guided Search Framework)* desenvolvida por Anderson *et. al.* [Anderson et al. 2000, Scott, Lesh e Klau 2002, Lesh, Marks e Patrignani 2000], permite ao usuário e ao método computacional

trabalharem de modo bem definido: o algoritmo é responsável por procurar uma solução inicial de boa qualidade para um problema de otimização, enquanto, o usuário tem a tarefa de ajudá-lo a resolver sub-casos particulares do problema, executar mudanças manuais na solução existente e recuperar uma solução anterior. Cabe ao homem ainda, a tarefa de tirar o algoritmo de mínimos locais e guiá-lo para a solução ótima. As ações que o homem pode executar na *HuGS* são:

- editar uma solução;
- focalizar o método em regiões do espaço de pesquisa, determinando níveis de prioridade;
- recuperar as soluções anteriores e;
- invocar, controlar e parar os métodos de otimização.

A *HuGS* foi utilizada com sucesso para resolver um problema de *CVRTW (Capacitated Vehicle Routing with Time Windows)* [Anderson et al. 2000], desenho de grafos [Lesh, Marks e Patrignani 2000] e escalonamento de tarefas [Lesh et al. 2000].

Uma abordagem semelhante à *HuGS* é o *User Hints Framework* apresentado por Nascimento [Nascimento 2003, Nascimento e Eades 2001]. Como no *HuGS*, esta abordagem permite ao usuário guiar o método de otimização. A principal diferença está no fato de que o *User Hints* suporta a inclusão de novos domínios de conhecimento de forma dinâmica, trabalhando com ambos os aspectos: convergência e refinamento da solução. Consequentemente, o usuário pode, em tempo de execução, modificar os objetivos e as restrições do problema, a fim de refiná-lo ou de incluir novas informações à medida que elas se tornem disponíveis.

Investigações com a abordagem *User Hints* mostraram que a interação homem-computador é benéfica para alguns processos de otimização complexos [Nascimento 2003]. Sendo possível através da junção das habilidades e conhecimentos humanos com o poder computacional encontrar uma solução melhor para problemas de otimização cuja a resolução baseada somente em métodos computacionais não alcança o objetivo desejado.

Utilizando a abordagem *User Hints* Nascimento implementou alguns sistemas de otimização, entre eles, uma aplicação interativa para desenho de grafos direcionados, denominada *GDHints*. Esse sistema permite que um usuário auxilie o método de otimização de desenho de grafos. Bons resultados foram obtidos, os usuários que testaram a aplicação, em sua grande maioria, conseguiram produzir desenhos que possuiam uma qualidade superior, em termos dos critérios estéticos estabelecidos, quando comparados aos desenhos criados pelo método de otimização trabalhando sozinho.

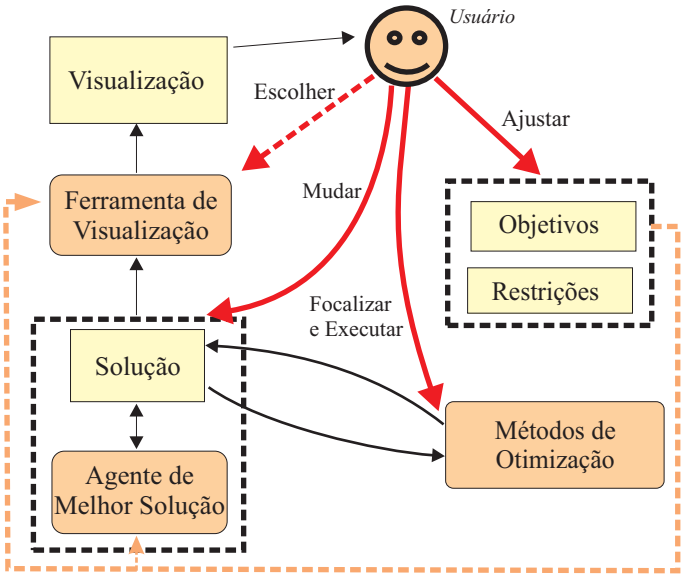


Figura 3.2: *Estrutura da abordagem* User Hints *[Nascimento 2003].*

A próxima seção descreve a abordagem *User Hints* e a forma como ela trata o processo de otimização.

3.2.1 Abordagem *User Hints*

O processo de otimização na abordagem *User Hints* possui três soluções: a **solução inicial** que é gerada automaticamente através dos métodos no início da otimização; a **solução de trabalho**, a qual é baseada na primeira e que pode ser manipulada pelo usuário; e a **melhor solução**, atualizada automaticamente toda a vez que a solução de trabalho supera a sua qualidade.

Estrutura da Abordagem

A Figura 3.2 mostra a estrutura da abordagem *User Hints* que é composta de nove elementos. A conexão entre eles é representado por setas, sendo que as linhas pontilhadas indicam uma dependência implícita. Os elementos são:

- **Usuário:** é responsável por refinar o problema e guiar o método de otimização na convergência para a solução ótima;
- **Objetivos:** descreve o custo (ou qualidade) de uma solução segundo a definição tradicional de um problema de otimização;
- **Restrições:** são condições impostas às variáveis do problema. São modeladas antes do processo de otimização começar, mas podem ser alteradas pelo usuário dinamicamente;

- **Solução:** é a solução de trabalho que será refinada pelo usuário até obter um resultado final de boa qualidade;
- **Métodos de otimização:** este módulo contém algoritmos tradicionais para resolver o problema de otimização estudado. Heurísticas específicas, meta-heurísticas e métodos exatos podem ser disponibilizados;
- **Função de qualidade:** é um elemento que não está representado no diagrama da Figura 3.2. Essa função é semelhante à função objetivo, incluindo apenas um custo adicional para a não satisfação de restrições do problema. Em geral, algumas restrições que são muito difíceis de serem satisfeitas são flexibilizadas e um custo associado a elas é incluído na função qualidade. As soluções geradas pelos métodos de otimização e pelo usuário são avaliadas segundo a função qualidade, e não pela função objetivo;
- **Agente de melhor solução:** é responsável por medir a qualidade da solução de trabalho e guardar a melhor solução assim que ela surge;
- **Ferramenta de visualização:** ponto de interação do usuário com o processo de otimização. É através da ferramenta de visualização que o usuário recebe a resposta sobre o processo de otimização. As visualizações são utilizadas também como mecanismos de entrada de dados. Através delas é que o usuário executa várias ações, como alterar manualmente a solução de trabalho, selecionar parte da solução, executar os métodos de otimização e alterar a definição do problema (sua função objetivo e restrições);
- **Visualização:** visualização da solução gerada.

Durante a otimização, o usuário pode efetuar operações em sua solução de trabalho que o auxiliará a inserir restrições que não foram modeladas e que são frutos do conhecimento de domínio do usuário, ajudar o método a escapar de mínimos locais e reduzir o espaço de solução do problema. Essas operações são chamadas *"dicas"*. Nascimento, em seu trabalho, enumerou três tipos de dicas diretas e três tipos de dicas indiretas. As dicas diretas são:

1. **Ajustes da função objetivo e restrições:** a qualquer momento, o usuário pode inserir ou remover restrições às variáveis do problema, bem como mudar a sua função objetivo;
2. **Focalização do método de otimização:** o usuário pode focalizar a ação do algoritmo em pontos específicos do problema. Isso é possível através da seleção de parte das variáveis do problema e execução de um método de otimização;

3. **Mudanças manuais:** são as modificações diretas que os usuários podem fazer nos valores das variáveis que compõem a solução de trabalho.

São consideradas dicas indiretas:

1. **Recuperar a melhor solução:** a qualquer momento a melhor solução pode ser recuperada e indicada como solução de trabalho;

2. **Atualizar a melhor solução:** ao contrário da dica anterior o usuário pode gravar a solução de trabalho como melhor solução da otimização;

3. **Controlar o módulo de otimização:** vários métodos de otimização podem ser implementados dentro da abordagem. O usuário pode escolher qual método executar, além de poder, a qualquer momento, parar um método em execução.

O processo de otimização dentro da abordagem *User Hints* começa com a geração automática de uma solução de trabalho inicial. O agente de melhor solução armazena então essa como a melhor solução da otimização.

A partir deste ponto, o usuário comanda a otimização, dando dicas ao sistema e re-executando os métodos de otimização. Qualquer modificação faz com que o agente de melhor solução seja notificado. O agente então compara a solução de trabalho com a melhor solução da otimização, se a solução de trabalho tiver uma qualidade superior ela é salva como a nova melhor solução da otimização.

Através de muitas interações, onde a intervenção humana se alterna com re-execuções do algoritmo de otimização, uma melhor solução para o problema é gradualmente produzida.

Todas as modificações que podem ocorrer na solução de trabalho são frutos das dicas do usuário ou de modificações automáticas dos métodos de otimização.

A abordagem *User Hints* pode ser utilizada em qualquer problema de otimização em que o usuário, por sua experiência e conhecimentos anteriores, é capaz de auxiliar na obtenção de melhores soluções. Em [Nascimento 2003] são descritos três experimentos com a abordagem *User Hints*, dentre os quais se encontra uma aplicação para problemas de desenho de grafos direcionados.

A área de desenho de grafos contém uma grande quantidade de heurísticas que satisfazem vários critérios estéticos. Alguns algoritmos trabalham apenas com um subconjunto de critérios, desconsiderando outros. Estes critérios podem ser conflitantes e, em caso de conflito, é difícil determinar qual deles deve ser prioridade. Devido a estes fatores os critérios estéticos para desenho de grafos são associados a problemas de otimização que são difíceis computacionalmente de serem resolvidos [Battista et al. 1999].

O que acontece, na maioria das vezes em que um bom desenho de grafo é produzido, é a criação de uma solução inicial utilizando alguma heurística e, em seguida, um pós-processamento manual visando a melhorar os critérios estéticos pré estabelecidos para aquela solução [Nascimento 2003].

Devido à inerente necessidade da presença humana no processo de otimização de desenho de grafos, Nascimento investigou a construção de desenho de grafos de forma interativa. Ele adequou a abordagem *User Hints* para tal fim e desenvolveu uma aplicação interativa para desenho de grafos direcionados denominada *GDHints* [Nascimento 2003] utilizando o método Sugiyama de otimização (mais detalhes sobre esse algoritmo em [Battista et al. 1999]) e um algoritmo genético baseado em restrições e foco desenvolvido pelo próprio Nascimento [Nascimento e Eades 2002].

O sistema *GDHints* incluiu os seguintes elementos:

- uma *interface* gráfica, na qual o usuário visualiza o seu desenho de trabalho e tem a sua disposição ferramentas que lhe permitem focalizar a ação dos métodos em parte do desenho de grafo, adicionar e/ou remover restrições ou executar mudanças manuais na própria solução corrente;
- métodos de desenho de grafo e;
- exibição das métricas de qualidade do desenho.

Apesar do usuário poder alterar dinâmicamente as restrições na abordagem *User Hints*, nessa aplicação, durante o processo de otimização, as restrições de carater visual estético, que fazem parte do problema de otimização, não podem ser modificadas pelo usuário, ou seja, os usuários não podem impor novos critérios estéticos, remover ou mudar a prioridade dos critérios já estabelecidos. A aplicação *GDHints* trabalha principalmente com os critérios: minimização de cruzamento de arestas, minimização de vértices falsos, minimização de curvas em arestas e orientação das arestas.

Existe, contudo, a possibilidade dele inserir suas próprias restrições, mas estas restições são relacionadas a posicionamento de vértices em relação a outros (esquerda-direita ou cima-para-baixo).

Os experimentos com a abordagem *User Hints*[Nascimento 2003] para problemas de desenhos de grafos direcionados foram feitos por Nascimento utilizando a aplicação *GDHints*. Sendo que, bons resultados foram alcançados quando um usuário interagia simultaneamente com os métodos de otimização.

Entretanto, as abordagens para otimização interativa criadas até o momento que, focalizaram também problemas específicos de desenho de grafos, trataram esse problema envolvendo apenas uma única pessoa no processo de otimização. Visando ampliar os estudos em otimização interativa, este trabalho investiga a otimização

interativa de desenho de grafos com múltiplos usuários. O objetivo é estudar formas cooperativas de interação para resolução de problemas de otimização de um desenho de grafo e verificar os efeitos no processo de otimização com a inserção de mais de uma pessoa.

CAPÍTULO 4

Trabalho Colaborativo Suportado por Computador (CSCW)

Neste capítulo são apresentados os fundamentos a respeito de trabalho colaborativo e os fatores que influenciam a colaboração suportada por computador.

4.1 Trabalho Colaborativo

Um trabalho colaborativo (ou cooperativo) é aquele executado através da interação entre um grupo de pessoas co-localizadas[1] ou remotas, simultaneamente ou não [Grudin 1994, Hofte 1998]. Tarefas complexas, de caráter criativo, que exigem o conhecimento de diversas áreas ou tomadas de decisão são exemplos típicos de atividades que podem ser resolvidas de forma colaborativa.

A necessidade de entender como grupos cooperam e como são afetados pela tecnologia surgiu em meados dos anos 70. Nessa época existiam os "*office automation*"[Grudin 1994] que tentavam integrar editores de textos e planilhas eletrônicas para atividades em grupo dentro das empresas. Essa tentativa esbarrou em vários empecilhos, inclusive tecnológicos, mas o principal era o fato de os desenvolvedores não compreenderem os reais requerimentos dos sistemas [Grudin 1994]. Mais tarde, Bannon e Schmidt [Bannon e Schmidt 1991] redefiniram um termo já conhecido, CSCW, como "um esforço no sentido de entender a natureza e as características do trabalho cooperativo com o objetivo de projetar tecnologias computacionais adequadas".

Na verdade, a sigla CSCW (*Computer Supported Cooperative Work*) surgiu em 1984, em um *workshop* organizado por Iren Grief e Paul Cashman, nos Estados Unidos, quando cerca de vinte pesquisadores se reuniram para discutir como as pessoas trabalham juntas e como a tecnologia poderia auxiliá-las [Grudin 1994].

[1]Por pessoas co-localizadas entende-se aquelas que encontram-se em um mesmo ambiente físico, durante a execução de uma atividade colaborativa.

Existe muita discussão sobre o significado da sigla CSCW. O CW pode ser compreendido de diversas maneiras: *cooperative work, collaborative work, coordinated work* e *collective work* [Bannon e Schmidt 1991]. As definições "cooperativo" e "colaborativo" são as mais usadas. Grudin [Grudin 1994] estabeleceu as diferenças entre esses dois sentidos:

- ***Computer Suported Cooperative Work***: é aplicado a pequenos grupos de pessoas que possuem objetivos comuns e a necessidade de comunicação para alcançá-los. Para Ehn [Ehn 1988] *apud* [Bannon e Schmidt 1991] o trabalho cooperativo é aquele cujo sucesso é essencialmente dependente do sucesso individual de todos os usuários. Cooperação é inerentemete "com" outros, trabalhando juntamente com alguém [Grosz 1994], ou seja, o trabalho é executado em conjunto e com a participação de todos. A contribuição de cada membro do grupo é importante para que o objetivo final da atividade seja alcançado.
- ***Computer Suported Collaborative Work***: é aplicado a grupos maiores que possuem conflitos entre suas metas individuais. Em atividades colaborativas os objetivos podem ser distintos e independentes.

No restante deste texto o termo "colaborativo" será usado de maneira geral, ou seja, quando os conceitos se aplicarem a qualquer tipo de trabalho em grupo (colaborativo ou cooperativo).

4.1.1 Os Elementos do CSCW

Os estudos sobre CSCW definiram **interação**, **espaço** e **tempo** como os elementos fundamentais na definição dos padrões de relacionamento entre as pessoas envolvidas em alguma atividade colaborativa.

O fato de o grupo possuir um objetivo comum ou necessitar se comunicar para resolver algum problema, pressupõe que existirá uma **interação** entre eles. A interação é vista como um dos principais elementos de uma atividade colaborativa, em especial a interação face-a-face, a qual é inerente às atividades onde as pessoas estão co-localizadas, podendo utilizar expressões não-verbais para se comunicarem. Atos como apontar, expressões da face e olhares são ações típicas que ocorrem quando duas ou mais pessoas estão co-localizadas, em alguma tarefa [Hofte 1998, Ellis, Gibbs e Rein 1991].

Em um trabalho colaborativo não existe a necessidade das pessoas estarem no mesmo ambiente físico ou realizarem a tarefa ao mesmo tempo. As tecnologias atuais permitem que as pessoas interajam de **espaços** remotos. O ambiente que

suporta esse tipo de atividade é chamado ambiente de trabalho distribuído e eles devem possuir requerimentos adicionais que auxiliem a interação entre os usuários, como por exemplo, canais de vídeo e voz [Reinhard et al. 1994].

O **tempo** de execução ou resposta das atividades de cada usuário podem acontecer, com relação as atividades dos outros membros do grupo, de maneira síncrona ou assíncrona. E-mail é um exemplo de trabalho assíncrono [Reinhard et al. 1994] onde a resposta do destinatário, não acontece de forma síncrona com o envio da mensagem pelo remetente.

Assim, as aplicações para CSCW são divididas em relação ao tempo e espaço, a Tabela 4.1 mostra esta relação. Em particular esta pesquisa focaliza as atividades co-localizadas face-a-face.

Tabela 4.1: *Tabela CSCW - Tempo x Espaço.*

	Mesmo Lugar	Lugares Diferentes
Mesmo Momento	Atividade face-a-face	Atividade distribuída síncrona
Momentos Diferentes	Atividade interativa assíncrona	Atividade distribuída assíncrona

Um outro elemento que merece destaque em CSCW é a separação no ambiente de trabalho dos dados públicos, privados e pessoais. Dados **públicos** são aqueles que podem ser acessados e modificados por todos os integrantes do grupo. As informações **privadas** são visualizadas somente pelo seu proprietário. Shen *et. al.* [Shen, Lesh e Vernier 2003] definem dados **pessoais** como sendo aqueles que são semi-privados, ou seja, visíveis pelos outros mas não acessíveis, desta forma, todos conseguem visualizar as informações pessoas de cada um mas não conseguem modificá-las ou copiá-las. Nessa dissertação espaço privado e público serão identificados como espaço individual e compartilhado, respectivamente.

4.1.2 Ambiente de Trabalho Colaborativo

Normalmente quando as pessoas se reúnem em um local para resolver um determinado problema costumam ficar por várias horas ou dias executando essa atividade. Stewart *et. al.* [Stewart, Bederson e Druin 1999] enumeram algumas características básicas que esse ambiente deve possuir:

- **Manutenção da persistência dos dados**: todas as informações e resultados que venham a surgir na colaboração devem ser mantidos para decisões e acessos futuros;

- **Acesso imediato ao conhecimento**: como várias pessoas estarão colaborando na resolução de um problema, existe uma diversidade de conhecimentos e o ambiente deve possibilitar a todos um canal de comunicação que permita a elas acessarem o conhecimento umas das outras;
- **Espacialidade do ambiente**: o grupo deve ter uma visão espacial de todos os elementos da colaboração.

A utilização de recursos tecnológicos na construção de ambientes para atividades em grupo que suprisse essas necessidades fez surgir os chamados *ambientes colaborativos ampliados*, onde as tarefas são executadas com o uso de recursos de rede, video-conferência, *wireless*, *wallscreens*, *touchscreen* e *notebooks*, entre outros dispositivos [Park 2003].

Todo esse aparato tecnológico de *hardware* é importante em atividades colaborativas, contudo, as características de *interface* e interação das aplicações que darão suporte as tarefas executadas no ambiente colaborativo são de importância fundamental. A criação de uma aplicação para um ambiente colaborativo está ligada a novos paradigmas, tanto de desenvolvimento quanto de *interface* com o usuário, que devem ser levados em consideração pelo desenvolvedor de uma aplicação *groupware*.

Nas próximas seções serão discutidas as características de uma aplicação colaborativa e os fatores cognitivos, sociais, motivacionais e políticos que devem ser levados em consideração quando se desenvolve uma aplicação desse tipo.

4.1.3 Aplicações para Trabalho Colaborativo (*Groupware*)

Durante muito tempo os termos CSCW e *groupware* foram vistos como sinônimos. Atualmente aceita-se que são na verdade duas linhas distintas, *groupware* está relacionado com o *software* para atividades colaborativas enquanto CSCW estuda os fatores humanos e computacionais envolvidos com o trabalho em grupo.

A palavra *groupware* é a junção de *GROUP* e *softWARE*. Ellis *et. al.* [Ellis, Gibbs e Rein 1991] interpreta o termo como sendo "sistemas computacionais que auxiliam grupos de pessoas engajadas em uma tarefa (ou objetivo) comum e que provêem uma interface para um ambiente compartilhado".

Em resumo, *groupware* define uma classe de *software* que são produzidos para dar suporte às atividades em grupo, com o objetivo de aumentar a produtividade e funcionalidade das atividades executadas de forma colaborativa com o auxílio do computador.

Segundo Grudin [Grudin 1994], diferentemente das aplicações para um único usuário, nas aplicações *groupware* é crucial a observação social, motivacional e

aspectos políticos do espaço de trabalho, sendo necessário enfatizar a possibilidade de comunicação entre as pessoas envolvidas.

Para envolver os usuário através da interação em uma atividade colaborativa, um conceito importante é o de *Awareness* (consciência, percepção). Segundo Kirsch-Pinheiro *et. al.* [Kirsch-Pinheiro, Lima e Borges 2001] "*awareness* é o conhecimento geral sobre as atividades e sobre o grupo. É o conhecimento sobre o que aconteceu, o que vem acontecendo, o que está se passando agora e o que poderá acontecer dentro das atividades do grupo, e sobre o próprio grupo, seus objetivos e sua estrutura". Em [Sohlenkamp 1998] *apud* [Kirsch-Pinheiro, Lima e Borges 2001] Sohlenkamp afirma que *awareness* "significa uma compreensão do estado total do sistema, incluindo atividades passadas, *status* atual e opções futuras" .

Dessa forma, o desenvolvedor e o projetista da *interface groupware* devem estar atentos sobre a prioridade da percepção de atividades individuais ou coletivas dentro da aplicação.

4.1.4 A Consciência do Espaço de Trabalho (*Workspace Awareness*)

Gutwin e Greenberg [Gutwin e Greenberg 1996] definem consciência do espaço de trabalho como o entendimento da interação de outra pessoa com o espaço de trabalho compartilhado. Essa consciência inclui o conhecimento de quem está no espaço de trabalho, onde eles estão trabalhando, o que estão fazendo e o que intencionam fazer. Segundo Greenberg *et. al.* [Greenberg, Gutwin e Cockburn 1996] tal consciência "é mantida através da fala, através da observação das ações dos outros no espaço de trabalho, através de comunicação gestual, através de referências diretas e por observação da direção do olhar", de uma forma geral, está relacionado a posição, postura, movimento da cabeça, olhos e braços [Inkpen et al. 2001].

A consciência do usuário sobre as atividades individuais e coletivas é um fator crucial em uma colaboração pois, através do modelo mental que o usuário forma sobre todo o processo colaborativo é possível assegurar que as contribuições individuais serão relevantes para o grupo como um todo [Dourish e Bellotti 1992]. Segundo Gutwin e Greenberg [Gutwin, Roseman e Greenberg 1996] isso facilita a transição do usuário entre o espaço individual (privado e pessoal) e o compartilhado, sendo um fator importante na manutenção da usabilidade da *interface*.

Em ambientes cooperativos normais, composto por quadros, *flip-charts* e mesas, onde as pessoas estão co-localizadas, manter a consciência das pessoas sobre a atividade dos outros é mais fácil, uma vez que basta olhar para o lado e já se tem uma idéia do que o outro está fazendo. A visão da maioria das informações é

a mesma, principalmente aquelas exposta em murais. Entretanto, quando se fala de ambientes de trabalho virtuais e, principalmente, distribuído, manter a consciência do usuário é muito mais complicado.

Os sistemas *groupware* ainda não conseguem simular a riqueza de interações que os ambientes físicos oferecem [Greenberg, Gutwin e Cockburn 1996]. Eles limitam a área de visão, reduzem ou removem características de movimentos e sons e complicam a comunicação verbal [Gutwin, Roseman e Greenberg 1996].

Alguns autores [Gutwin, Roseman e Greenberg 1996, Dix 1997] estão propondo mecanismos de suporte à percepção e consciência para sistemas *groupware*. Esses mecanismos giram em torno de algumas questões que identificam aspectos vitais para o fornecimento de percepção dentro de um *groupware*. Essas questões foram compiladas da seguinte maneira:

- **O que**: refere-se ao conhecimento sobre o que cada usuário está fazendo, quais ferramentas está usando, qual a sua intenção, quais informações devem ser fornecidas a ele. Essa dimensão fornece a percepção ao usuário das atividades e papéis exercidos por cada membro dentro da colaboração;
- **Quando**: indica quando as mudanças serão feitas, ou seja, quando os eventos ocorrem e quando se dá apresentação das informações;
- **Onde**: refere-se à localização, visão e pesquisa da informação, onde as informações são geradas e apresentadas, onde os outros estão trabalhando, o que eles estão vendo, onde estão apontando;
- **Como**: é a informação de como as alterações aconteceram e como as informações serão apresentadas aos usuários;
- **Quem**: diz respeito à informação de quem está participando no momento;
- **Quanto**: qual a quantidade ideal de informação que deve ser apresentada ao usuário para que aconteça a percepção sobre o grupo e suas atividades;
- **Por quê**: refere-se ao contexto e a comunicação. Em conjunto com a questão "o que" permite saber porque uma ação aconteceu.

Gutwin e Greenberg [Gutwin e Greenberg 1996] organizaram algumas dessas questões em onze categorias que representam os elementos que, segundo eles, fazem parte do espaço de trabalho compartilhado e "prevêem um vocabulário básico para o pensamento sobre os requerimentos de consciência e suporte *groupware*." A Tabela 4.2 sumariza essa classificação.

Tabela 4.2: *Elementos da consciência do espaço de trabalho.*

Número	Elementos	Questões Relevantes
1	Presença	Quem está participando da atividade?
2	Localização	Onde eles estão trabalhando?
3	Atividades	Quais atividades eles estão fazendo?
4	Ações	O que eles estão fazendo? Quais são suas atividades e tarefas?
5	Intenções	Qual será o próximo passo deles? Onde eles estarão?
6	Mudanças	Quais mudanças eles estão fazendo e onde?
7	Objetivos	Quais objetos eles estão usando?
8	Extensão	O que eles podem ver?
9	Habilidades	O que eles podem fazer?
10	Esfera de influência	Onde eles podem fazer mudanças?
11	Expectativas	O que eles precisam fazer no próximo passo?

Com estas questões em mente algumas considerações importantes sobre a consciência do espaço de trabalho devem ser observadas durante o projeto de uma *interface groupware* [Gutwin, Roseman e Greenberg 1996, Gutwin e Greenberg 1996]:

- A *interface* deve cobrir o conhecimento do estado do espaço de trabalho, de seus artefatos e das ações de cada usuário dentro desse contexto;
- A *interface* deve ser facilmente entendida, principalmente para evitar distrações que tem a ver com dificuldade de interpretação;
- A consciência da informação deve ser coletada e distribuída pelo sistema em lugar de explicitamente procurada pelos participantes.

Segundo Kirsch-Pinheiro [Kirsch-Pinheiro, Lima e Borges 2001], apesar da importância fundamental da consciência do espaço de trabalho, "os usuários não podem se sentir soterrados pelas informações de percepção, nem tê-las omitidas. Estas informações também devem ter uma natureza passiva, surgindo direto das

atividades de cada pessoa, ao invés de ser gerenciada ou procurada explicitamente. Uma *interface* mal projetada pode causar tanto a sobrecarga, quanto a omissão, por ser inadequada ao tipo de informação apresentada".

A próxima seção discute os elementos de *interface* que podem ser inseridos em uma aplicação *groupware* e auxiliam a manutenção da consciência do usuário sobre o espaço de trabalho.

4.1.5 *Interface* das Aplicações *Groupware*

A criação da *interface* para aplicações *groupware* deve incluir técnicas para ambas as interações homem-computador e homem-homem, além de técnicas para visualização de informação.

A visualização compartilhada da aplicação é um fator dificultante em aplicações *groupware* e algumas regras de *design* podem ser utilizadas para minimizar essa dificuldade. Elas envolvem desde a visão individual até técnicas para a visualização compatilhada. Dias [DIAS 1998] *apud* [Kirsch-Pinheiro, Lima e Borges 2001] em seu trabalho apresenta quatro modelos de *interfaces groupware* possíveis, elas estão ligadas principalmente a aplicações onde os usuários estão distribuídos no espaço, mas podem também ser utilizadas em sistemas para usuários co-localizados:

- WYGIWIG (*What You Get Is What I Get*): os usuários possuem a mesma visão uma vez que compartilham as mesmas janelas. Contudo, pequenas inconsistências, coordenação e atrasos são tolerados se não comprometerem as informações que podem ser recuperadas do espaço compartilhado pelos usuários;
- WYSIWID (*What You See Is What I Did*): segundo Kirsch-Pinheiro [Kirsch-Pinheiro, Lima e Borges 2001] "neste modelo, as interações entre os usuários se dão de forma semi-síncrona, com a propagação das alterações para os demais usuários podendo ser adiada até que condições preestabelecidas sejam satisfeitas, fazendo com que a visão que determinado usuário tenha das informações compartilhadas seja, por um curto espaço de tempo, defasada em relação à real situação atual";
- WYSIWIS (*What You See Is What I See*): definida por Stefik *et. al.* [Stefik et al. 1987]. Nesta abordagem a *interface* permite aos usuários compartilharem a mesma visão. O WYSIWIS é uma abstração que guia o desenvolvimento de *interfaces* para aplicações multiusuário baseadas nas propriedades de um quadro branco, por exemplo, onde todo o grupo tem a mesma visão das informações. Possui as vantagens de ser de simples implementação e fornecer um forte senso de contexto compartilhado. Por outro lado, não permitem

que usuários tenham espaço de trabalho privado, a tela pode ficar tumultuada com janelas que só estão sendo usadas por outros usuários e os vários cursores podem atrapalham o trabalho do grupo;

- WYSIAWIS (*What You See Is Almost What I See*): os usuários compartilham janelas, contudo são permitidas variações que não prejudiquem a interpretação dos resultados das ações dos outros usuários.

As *interfaces* WYSIWIS são as mais inflexíveis. Assim, para permitir interações que ocorrem em uma atividade colaborativa, pode-se relaxar esses tipos de *interface*. As aplicações WYSIWIS relaxadas podem ser criadas com um enfoque em quatro dimensões de relaxamento [Stefik et al. 1987]:

- **Espaço**: em uma aplicação WYSIWIS a visão de todos deve ser exatamente a mesma. Relaxando o elemento espaço, aplica-se o WYSIWIS apenas para alguns objetos visíveis mais importantes (janelas e cursores por exemplo);
- **Tempo**: enquanto em uma aplicação WYSIWIS as imagens devem ser sincronizadas, no relaxamento da dimensão tempo permite-se atrasos na atualização e visão simultânea entre os usuários;
- **População**: relaxando-se a dimensão população é possível permite que apenas um subgrupo de usuários compartilhe a mesma visão;
- **Congruência da visão**: o relaxamento dessa dimensão permite que usuários alterem sua visão local da aplicação sem que os outros sejam afetados.

Segundo Greenberg *et. al* [Greenberg, Gutwin e Cockburn 1996], "as relaxações dão um controle às pessoas da sua própria visão dentro do espaço de trabalho, permitindo a elas trabalharem de forma mais natural, mudando o foco entre o trabalho individual e coletivo. Entretanto, as aplicações WYSIWIS relaxadas podem contribuir para a perda de consciência, uma vez que, quando mudam a visualização, as pessoas podem perder a noção de onde os outros estavam e o que estavam fazendo no espaço de trabalho".

Para melhorar a interação das aplicações *groupware*, inclusive das aplicações WYSIWIS, a utilização de alguns controles multiusuários estão sendo estudados por pesquisadores. Gutwin e Greenberg [Gutwin, Roseman e Greenberg 1996] apresentam quatro elementos de tela (botões, caixas de texto, listas, etc.) para aplicações *groupware* distribuídas:

- **Visão em radar**: Radar é uma visão geral, em miniatura, do documento sobreposta por áreas coloridas que indicam o ponto de vista de cada usuário. Claramente percebe-se que esse tipo de controle privilegia os elementos da

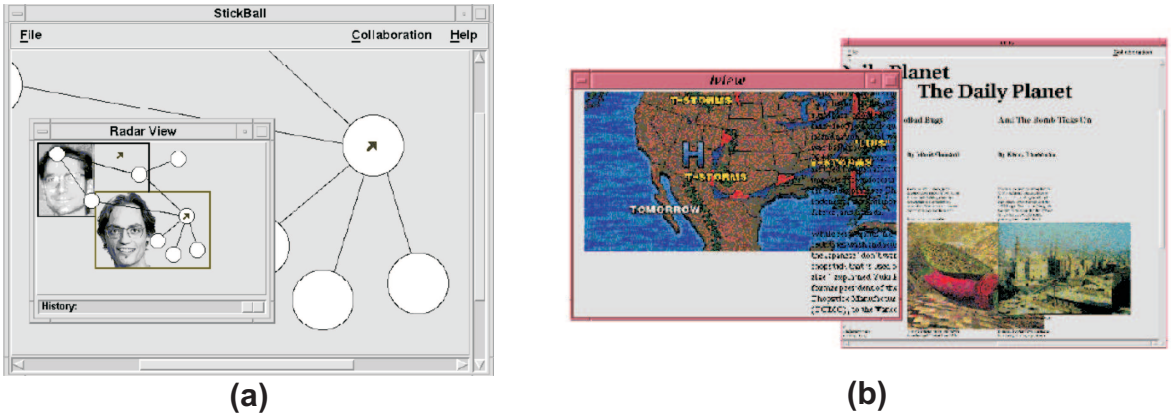


Figura 4.1: *(a) Visão em radar [Gutwin e Greenberg 1996] (b) Controle "what you see is what I do". A janela da esquerda mostra uma área limitada da visão do usuário remoto [Gutwin, Greenberg e Roseman 1996].*

consciência do espaço de trabalho *presença* e *localização*, conforme definido na Tabela 4.2. A letra (a) da Figura 4.1 mostra uma aplicação multiusuário com visão em radar para um desenho de grafo;

- **Visão em miniatura**: Miniaturas são espécies de ícones que mostram a visão de cada usuário. Esse tipo de controle também privilegia a *presença* e *localização*;
- **Barra de rolagem multiusuário**: são um conjunto de barras de rolagem, cada uma representando a posição de um usuário no documento compartilhado. Através da barra de rolagem multiusuário pode-se "seguir" um usuário, "colando" sua barra de rolagem à dele. O principal elemento de consciência inserido no espaço de trabalho através desse controle é a *localização*;
- **Visão WYSIWID**: provê detalhes, em tela cheia, da interação de outra pessoa. A letra (b) da Figura 4.1 mostra uma aplicação multiusuário com visão WYSIWID.

Toda a discussão nas seções anteriores serve para ambos os tipos de aplicações, distribuídas ou co-localizadas, mas com maior enfoque na primeira. Este trabalho baseia-se em um tipo de aplicação *groupware* denominada *Single Display Groupware* que focaliza colaboração co-localizada face-a-face e que será discutida na próxima seção.

4.1.6 *Single Display Groupware*

Stewart *et. al.* [Stewart, Bederson e Druin 1999] definem *Single Display Groupware* (SDG) como sendo "um programa de computador que permite usuários co-localizados colaborarem via um computador compartilhado com um único *display* e simultâneos dispositivos de entrada" [Stewart, Bederson e Druin 1999]. Este tipo de aplicação é utilizada quando um grupo de pessoas precisam interagir e trocar informação diretamente em uma mesma aplicação [Stewart, Bederson e Druin 1999].

Claro que as pessoas podem sentar ao redor do computador e trabalharem em uma aplicação normal mesmo tendo um único dispositivo de *mouse* e teclado acoplados a máquina. Contudo, nessa situação o normal é que um dos participantes do grupo controle o computador, enquanto os demais interagem com o sistema de forma indireta.

Estudos feitos com pares de crianças [Scott, Mandryk e Inkpen 2003] nesse tipo de configuração mostraram que, em geral, apenas uma das mesmas manipulava o dispositivo. Enquanto, a outra não interagia com a máquina. Três padrões diferentes de comportamento das crianças foram verificados:

- as duas crianças trabalhavam em conjunto na execução da atividade e todos os passos eram feitos em comum acordo;
- a criança que não possuía o controle do *mouse* interagia com a aplicação apontando na tela do computador e gerenciando a cooperação. Ela era o elemento ativo da cooperação e a outra apenas recebia ordens;
- a criança que detinha o controle do *mouse* executava a atividade, e a outra ficava apenas como observadora do processo, por vezes até perdendo o interesse e executando outra tarefa.

Essas três situações podem ocorrer em qualquer trabalho em grupo auxiliado por computador. Contudo, as duas últimas, que são menos colaborativas não são desejadas. Uma forma de resolvê-las é possibilitar que todos utilizem a aplicação usando dispositivos de entrada individuais, por exemplo, um *mouse* próprio.

O uso de programas SDG em trabalhos cooperativos possui vários benefícios. Stewart *et. al.* [Stewart, Bederson e Druin 1999] enumeram algumas dessas vantagens:

- habilita a cooperação que foi inibida por barreiras sociais;
- permite atividades que são naturalmente executadas por múltiplos usuários;
- aumenta as possibilidades na utilização de um computador em uma cooperação;

- reduz ou elimina conflitos quando múltiplos usuários interagem simultaneamente com uma única aplicação e único dispositivo de entrada e;
- possibilita o aprendizado cooperativo.

Entre as desvantagens, também segundo Stewart *et. al.*:

- podem acontecer conflitos entre os usuários quando eles executam ações incompatíveis, simultaneamente;
- pode haver a diminuição da colaboração pela possibilidade do trabalho ser executado em paralelo;
- o espaço reduzido da tela de um monitor normal pode dificultar a interação entre os usuários;
- as aplicações SDG costumam ser mais lentas que as aplicações normais e;
- a aplicações SDG são dependentes dos mais populares sistemas operacionais e, assim, podem não ser portáveis.

Ainda não existem soluções comerciais para o desenvolvimento de aplicações SDG. Os protótipos construídos até o momento são estudos feitos pela comunidade científica que se mostram por vezes lentos e de difícil interação. Alguns pesquisadores envolvidos na construção de aplicações colaborativas afirmam que o desenvolvimento de uma aplicação para múltiplos usuários não é algo fácil de ser implementado. Existem vários aspectos que envolvem a modelagem de uma aplicação SDG que precisam ser considerados.

O problema começa com a utilização de múltiplos dispositivos de entrada. Os sistemas operacionais atuais não dão suporte natural a utilização de vários apontadores e teclados ao mesmo tempo. Alguns até reconhecem a existência de mais de um destes, mas não exibem e rotulam, por exemplo, apontadores individuais na tela. Essa funcionalidade precisa ser implementada pelo desenvolvedor que, além de construir as rotinas que identificarão os múltiplos dispositivos no sistema deverá, estar atento às seguintes questões: como identificar qual é o ponteiro ou teclado de cada usuário? Como identificar as ações de cada usuário individualmente no sistema? Como possibilitar que a ação de um usuário não interfira na atividade de outro?

Uma outra dificuldade é a *interface*. Alguns dos elementos de tela para aplicações *groupware* foram desenvolvidos [Greenberg, Gutwin e Cockburn 1996, Gutwin, Roseman e Greenberg 1996] (ver Seção 4.1.5). Contudo, existem diferenças na manipulação dos controles (botões, menus, etc.) e na realização de tarefas comuns entre uma *interface* mono e multiusuário. Os controles para aplicações monousuário podem ser selecionados somente um a cada vez e por vezes em certa ordem (por

exemplo, copiar e colar), dependendo do contexto da aplicação determinado controle não pode ser acionado.

Em caso de aplicações multiusuário, a situação não é tão simples, pois os usuários podem estar em modo de trabalho diferentes e usando funções distintas ao memo tempo. Um exemplo típico de dificuldade envolve determinar e implementar a semântica das ações "desfazer" e "refazer" em um ambiente SDG. Neste caso, o "desfazer" pode reverter a última ação feita pelo usuário que invocou ou a última ação realizada por qualquer usuário.

Zanella e Greenberg [Zanella e Greenberg 2000] identificaram alguns cuidados que uma aplicação SDG deve ter, relativos à interação com os elementos de tela. Esses cuidados foram resumidos na lista abaixo:

- identificar o dispositivo e o trabalho de cada usuário;
- prover elementos de tela que reconheçam e respondam a ações de usuários diferentes;
- desenvolver uma *interface* que evite a interferência de um usuário na atividade do outro;
- prover a consciência do espaço de trabalho;
- definir uma interface que facilite a interação entre os usuários;
- permitir a manipulação dos elementos de tela em sintonia com a posição e orientação do usuário e do dispositivo de saída.

Tse e Greenberg [Tse e Greenberg 2004, Tse e Greenberg 2002] desenvolveram um *toolkit* para auxiliar construção de aplicações colaborativas. O *SDGToolkit*[2], como foi chamado, permite a criação em ambiente .Net[3] de aplicações com suporte a múltiplos dispositivos de entrada (mouse e teclado), sem a necessidade de manipulação de baixo nível, ou seja, acesso direto ao *hardware*, desses dispositivos diretamente pela aplicação.

Além deste *SDGToolkit* Tse também desenvolveu alguns controles gráficos de tela para serem usados pelos dispositivos de entrada controlados pelo *SDGToolkit*. Esses controles além de poderem ser acionados por todos os dispositivos de entrada conectados ao computador têm a característica de serem multiusuário. Dessa forma, o acionamento de um controle gera uma ação individualizada para cada usuário.

[2]O protótipo desenvolvido nesta pesquisa utiliza essa API para dar suporte a múltiplos dispositivos de *mouse* em ambiente Windows NT/2000/XP.

[3]A plataforma .NET é um *framework* criado pela Microsoft para desenvolvimento, entre outros, de aplicações para *desktop*, para dispositivos móveis e aplicativos WEB. Permitindo uma integração entre várias formas de desenvolvimento e linguagens.

Além da possibilidade de identificação, através de cores, dos usuários que o acionaram.

Existem outras implementações que também facilitam a utilização de vários dispositivos de entrada simultâneos, mas que são menos "amigáveis" e não funcionam tão bem na plataforma Windows NT/2000/XP como o *SDGToolkit*. Entre elas se encontram:

- ***MAME:Analog+*** **[Mame 2006]**: uma biblioteca de jogos com suporte a múltiplos mouse;
- ***Multiple Input Devices - MID***: uma biblioteca genérica Java [Shoemaker 2000, Hourcade e Bederson 1999];
- ***CPNMouse*** **[Westergaard 2002]**: uma API escrita em C com utilização em aplicações para desenho de redes de Petri e jogos;
- **CIDA [Farias et al. 2006]**: uma ferramenta que permite a utilização de vários dispositivos de entrada através da utilização de *pluggins*.

Um outro estudo interessante foi o feito por Wallace *et. al.* [Wallace et al. 2004], visando estender o sistema *X-Window*[4] para suportar múltiplos cursores simultâneamente.

Além das características operacionais de desenvolvimento de aplicações SDG, aspectos sociais, como a privacidade da informação, devem ser levados em consideração na *interface*. Como exemplo, cada usuário pode ter acesso a sua própria informação particular, mas é preciso definir como o sistema controlará quem poderá ver e utilizar informações privadas. Em [Mandryk, Scott e Inkpen 2002, Zanella e Greenberg 2001, Shoemaker 2001] são apresentadas algumas soluções para essas questões.

A manutenção da consciência do espaço de trabalho também é importante em aplicações SDG. Contudo, boa parte dos estudos sobre consciência do espaço de trabalho vêem tendo um enfoque maior em sistemas *groupware* distribuídos [Shoemaker 2000], uma vez que, manter essa consciência em sistemas SDG é mais fácil, pois, os usuários possuem uma boa percepção do que os outros participantes estão fazendo por compartilharem a mesma visão.

4.1.7 Fatores que Influenciam a Colaboração Co-Localizada

Mandryk *et. al.* [Mandryk, Scott e Inkpen 2002] identificaram alguns fatores que influenciam a colaboração em um ambiente onde as pessoas estão co-

[4]O X-Window é uma *interface* gráfica para sistemas Unix, desenvolvido no MIT (*Massachussets Institute of Technology*) que permite criar janelas compatíveis com qualquer sistema operacional.

localizadas. A princípio esses fatores foram baseados em observações formais e informais. Durante pesquisas posteriores, Inkpen *et. al.* [Inkpen et al. 2005] fizeram experiências controladas sobre a real influência desses fatores em uma atividade cooperativa co-localizada específica. A seguir são detalhados os fatores e resultados desses estudos.

Orientação da tela

A orientação da tela (vertical ou horizontal) pode impactar fortemente a colaboração. Uma superfície horizontal permite aos usuários uma independência maior de posicionamento em relação a tela, possibilitando, assim, visualizar o problema em outra perspectiva, o que já não acontece com a superficie vertical, uma vez que, os usuários não possuem essa flexibilidade na mudança de visualização em relação a *interface*. Entretanto, a pesquisa de Mandryk *et. al.* mostra que uma tela orientada verticalmente costuma ser mais confortável para trabalhos que exigem por longos períodos de execução.

Disposição dos usuários

O fato de as pessoas estarem sentadas face-a-face ou lado-a-lado influência as interações entre as mesmas. A disposição lado-a-lado permite uma mesma visão da tela, entretanto, a disposição face-a-face se mostra melhor quando existe a necessidade de contato visual com a utilização de expressões não verbais.

Tamanho da tela

Quanto maior o tamanho da tela mais pessoas poderão participar da atividade. Em telas grandes é mais fácil mostrar e monitorar o que cada usuário está fazendo. Isso aumenta o poder de interação entre os usuários. Telas pequenas são usadas normalmente para atividades individuais e exibição de dados privados. Uma vantagem das telas pequenas é a correlação do tamanho da estrutura visual do problema com o campo de visão dos usuários.

Número de telas

As múltiplas telas permitem o particionamento da tarefa e amplia o controle do usuário sob o seu trabalho individual. Neste tipo de configuração os usuários tendem a focar mais em seu trabalho individual e interagem menos com os outros, enquanto que a tela compartilhada pode ampliar a atenção sob a atividade como um todo. Quando compartilham a mesma tela, os usuário tendem a ficar mais próximos

e se comunicarem mais. Contudo, em ambientes com múltiplas telas a coordenação e a manipulação de dados privados pode ser mais fácil.

Proximidade da tela

Quanto mais próximo o usuário estiver da tela, mais fácil será a sua interação direta com elementos da aplicação. Isso melhorará também a gesticulação, o que amplia a comunicação e a conversação entre os integrantes do grupo.

4.2 Otimização Interativa Multiusuário de Desenho de Grafos

A tarefa de encontrar o melhor desenho de um grafo é um problema de otimização combinatória cujo objetivo é otimizar um ou mais critérios estéticos, obedecendo restrições [Kaufmann e Wagner 2001].

Um problema de otimização combinatória é composto de variáveis, função objetivo e restrições (ver Seção 3.1). O problema de desenho de grafos possui todos estes elementos de um problema de otimização: os elementos do grafo (vértices e arestas) são as variáveis de entrada, os critérios estéticos compõem a função objetivo do problema, sendo possível ainda impor algumas restrições.

Tunkelang [Tunkelang 1999] define dois tipos de restrições: as semânticas e as interações do usuário. As semânticas interferem na estrutura do desenho, por exemplo, através da definição de um agrupamento de sub-vértices ou uma forma pré-definida para o desenho. Mas, os usuários poderiam definir suas próprias restrições fixando, por exemplo, o posicionamento de vértice em relação a outros (padrão esquerda-direita ou cima-baixo)[5] ou até determinar que um vértice específico deve ficar em uma posição fixa, como no centro ou fora do limites do desenho.

Tamassia [Tamassia 1998] enumera as principais restrições que podem ser impostas a um desenho de grafo:

- colocar um vértice específico no "centro" do desenho;
- colocar um vértice específico no limite do desenho;
- colocar um conjunto de vértices juntos;
- colocar um caminho horizontalmente alinhado da esquerda para direita (ou verticalmente do topo para baixo) e;

[5] O padrão esquerda-direita determina que um vértice deve ficar sempre a esquerda ou direita de outro. Já o modelo cima-baixo assume que um vértice específico deve se localizar sempre acima ou abaixo de outro.

- desenhar um subgrafo com uma forma específica.

Já as preferências são mais fracas que as restrições. São parâmetros desejados e descrevem uma idéia do usuário a respeito do desenho. Tunkelang [Tunkelang 1999] exemplifica a diferença entre uma restrição e uma preferência, afirmando que pode-se impor a restrição de que um vértice deve ficar a uma determinada distância de outro, ou designar uma distância preferencial entre dois vértices, com possibilidade de ajustes se necessário. As preferências são, na verdade, os critérios estéticos discutidos da Seção 2.2.1.

Para um método de otimização, as preferências são modeladas como a função objetivo do sistema. Essa função tem as posições dos vértices como entrada e retorna uma medida numérica. Por exemplo, para um grafo G poderíamos ipoteticamente construir uma função objetivo $f(G)$ como a descrita abaixo:

$$f(G) = c_1 \times \alpha(G) + c_2 \times \frac{1}{\delta(G)} + \ldots \tag{4-1}$$

Sendo $\alpha(G)$ o número de cruzamentos existentes no grafo G, $\delta(G)$ o desvio padrão do tamanho da maior aresta e c_i, $i = 1, 2, \ldots$ constantes.

Mas, tanto para as restrições quanto para as preferências, o usuário poderá, caso este não esteja satisfeito com a solução apresentada pelo método, modificá-las e re-computar uma nova solução. O que dá um caracter interativo à otimização de um desenho de grafos.

Conforme as seção anteriores, foram feitos alguns estudos na área de otimização interativa de desenho de grafos envolvendo um único usuário no processo de otimização, como os trabalhos de Anderson [Anderson et al. 2000] e Nascimento [Nascimento 2003]. Contudo, via de regra, usuários distintos geralmente tem percepções diferentes sobre como melhorar um desenho. É portanto interessante investigar formas de permitir cooperação entre usuários na atividade de desenho de grafos.

Até o momento só se tem conhecimento nesse sentido do trabalho desenvolvido por Kobourov e Pitta [Kobourov e Pitta 2004]. Eles implementaram uma aplicação multiusuário para otimização interativa de desenho de grafos que utiliza a abordagem *HuGs*.

Essa aplicação trabalha com desenhos simultâneos de grafos. Recebe como entrada dois grafos distintos que possuem o conjunto de vértices e conjuntos diferentes de arestas. Dois desenhos são gerados e a disposição dos vértices na tela é a mesma para ambos. À medida que um vértice é movido em um desenho, o correspondente no outro é automaticamente atualizado para a mesma posição. O objetivo é produzir, simultâneamente dois desenhos que possuam poucos ou nenhum cruzamento de arestas e simetria. Caso os dois grafos sejam planares, a meta é

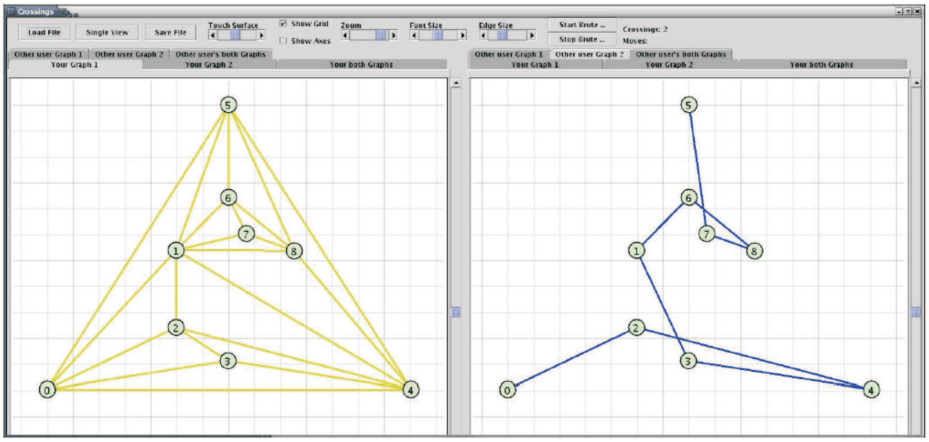


Figura 4.2: *Sistema interativo para desenho de grafos [Kobourov e Pitta 2004].*

conseguir que os respectivos desenhos sejam planares também. A Figura 4.2 ilustra a tela da aplicação.

Segundo Kobourov e Pitta através da visualização conjunta dos dois desenhos é possível comparar o relacionamento que existe entre eles, como por exemplo, visualizar árvores evolucionárias de um mesmo conjunto de espécies. Essa aplicação permite a interação de mais de um usuário.

Este trabalho, contudo, não tem enfoque no problema multiusuário de desenho de grafos. Sua característica cooperativa se justifica apenas pela possibilidade da interação explícita de mais de um usuário no problema. O objetivo foi provar que determinados pares de grafos possuíam desenhos simultâneos com algumas características estéticas, para isso Kobourov e Pitta utilizaram algoritmos de otimização e a interação humana.

Em contra-partida o presente trabalho visa investigar a resolução cooperativa de um desenho de grafo para verificar se a proximidade dos usuários pode ampliar a interação auxiliando a descoberta da melhor solução, o que será feito a partir do próximo capítulo.

CAPÍTULO 5

Abordagem *User Hints* Cooperativa - Uma Extensão da Abordagem *User Hints*

Este capítulo apresenta uma abordagem interativa e cooperativa para a resolução de problemas de otimização complexos, denominada *Co-UserHints*. A proposta é uma extensão do modelo original do *User Hints Framework*.

5.1 Modelos de Otimização Interativa Cooperativa

A abordagem *User Hints* [Nascimento 2003], discutida no Capítulo 3, modela a interação entre um único usuário e o computador. Contudo, problemas complexos de otimização podem ser resolvidos de maneira mais eficiente se houver a interação de vários usuários em conjunto com métodos semi-automáticos. Essa interação permite a divisão de tarefas, a ampliação do conhecimento localizado, a discussão de pontos de vista diferentes, a solução de conflitos, a elaboração de estratégias para a resolução de um problema, o controle e a revisão de decisões e em alguns casos a diminuição no tempo de execução, entre outras atividades.

O planejamento de distribuição de energia é um exemplo prático de um problema de otimização complexo [Khator e Leung 1997] que pode ser resolvido interativamente através do trabalho cooperativo. Ele consiste em decidir como atender uma demanda de carga através da utilização de subestações elétricas e alimentadores de energia. A sua resolução inclui muitas decisões estratégicas envolvendo diversos componentes de uma rede de energia elétrica e restrições ambientais como, por exemplo: a localização das subestações e dos alimentadores, a alocação de carga e a alocação de capacidade de cada subestação. Dada a sua complexidade, um grupo de engenheiros poderia resolvê-lo de forma cooperativa usando a sua inteligência coletiva unida ao poder de pré-processamento de um sistema computacional.

De um modo geral é possível identificar três formas básicas de se resolver um problema de otimização com um grupo de pessoas:

- **Trabalho individual em soluções completas**[1]**:** nessa configuração as pessoas trabalham em soluções individuais completas que são depois analisadas e integradas para gerar uma única solução. A integração das soluções individuais pode ser feita de forma manual (por ação dos usuários) ou de modo automático. Na integração automática o próprio sistema se encarrega de gerar a melhor solução através da análise e integração das soluções individuais. Esse modelo possui a vantagem de possibilitar aos usuários uma capacidade maior de concentração (sem interferência externa).
- **Trabalho cooperativo em uma única solução completa:** nesse modelo existe uma única solução do problema em que os usuários trabalham conjuntamente. Ele possui como vantagem a ampliação da possibilidade de cooperação natural entre os usuários, uma vez que permite a discussão de pontos de vista e elaboração de estratégias coletivas. O modelo é mais adequado se os usuários estiverem co-localizados, contudo, também pode ser implementado através de ambientes distribuídos.
- **Trabalho cooperativo em soluções parciais:** nesse modelo o grupo se beneficia de uma divisão explicita das tarefas onde cada usuário fica com uma parte do problema. Tal configuração é interessante quando existe a necessidade de atividades em paralelo, ou há partes do problema que podem ser executadas por pessoas especializadas. O sistema fica responsável por reunir os resultados (ou soluções) parciais geradas pelos usuários, a fim de produzir uma solução completa final.

A nova abordagem *User Hints* Cooperativa (*Co-UserHints*) para problemas de otimização interativa, descrita nas próximas seções, estende a proposta original do *User Hints Framework* para suportar cooperação com múltiplos usuários nas duas primeiras formas descritas acima.

Essa escolha foi baseada no interesse em se investigar como seria a colaboração entre usuários em ambientes com visualizações individuais ou visualizações coletivas do problema.

A terceira forma de cooperação por exigir a implementação de métodos avançados de integração das soluções parciais, principalmente no caso de desenho de grafos e, por falta de tempo, não foi avaliada.

Esta dissertação apresenta então uma abordagem *Co-UserHints* diferenciada para cada uma das configurações que serão analisadas. A primeira configuração

[1]Denomina-se solução completa aquela que envolve todo o problema e não apenas sub-casos do mesmo.

é considerada dentro de um modelo chamado *modelo coletivo*; a segunda é utilizada no *modelo integrado*.

Além disso, o presente estudo investiga somente ambientes onde os usuários estão co-localizados e trabalhando simultaneamente, utilizando aplicações SDG. Contudo, a abordagem *Co-UserHints* pode ser aplicada a ambientes de trabalho colaborativo distribuídos.

5.2 Abordagem *Co-UserHints*

A abordagem *Co-UserHints* possui todos os módulos do *User Hints Framework* descrito na Seção 3.2.1, mas modifica a funcionalidade de alguns de seus elementos e inclui novos recursos para cooperação entre múltiplos usuários.

Como na abordagem *User Hints*, o *Co-UserHints* implementa uma única função[2] objetivo que descreve a qualidade da solução de um problema. Contudo, essa função pode ser alterada dinamicamente por qualquer usuário. Além disso, a nova abordagem prevê dois tipos de restrições do problema: restrições globais, que afetam as soluções geradas por todos os usuários e, restrições locais associadas a cada usuário individualmente. Um usuário não pode alterar as restrições locais definidas por outros usuários. Entretanto, pode alterar as restrições globais caso haja permissão do grupo. Note que as restrições globais e locais podem ser conflitantes entre si. Na Seção 5.3.4 são descritas estratégias para resolução de conflitos entre restrições.

A semântica dos métodos de otimização é a mesma utilizada na abordagem *User Hints*. Convém ressaltar, entretanto, que na *Co-UserHints*, qualquer usuário pode acionar um método de otimização.

Outros elementos como o agente de melhor solução, a solução de trabalho, a função de qualidade e o módulo de visualização diferem nas abordagens dependendo do modelo de colaboração utilizado (coletivo ou integrado).

A seguir, a abordagem *Co-UserHints* no modelo coletivo é apresentado em detalhes.

5.3 *Co-UserHints* no Modelo Coletivo

No modelo coletivo os usuários podem trabalhar em dois níveis: *individual* e *cooperativo*, conforme ilustrados na Figura 5.1.

[2]Nesse modelo não foi investigado a possibilidade de uma função objetivo para cada usuário.

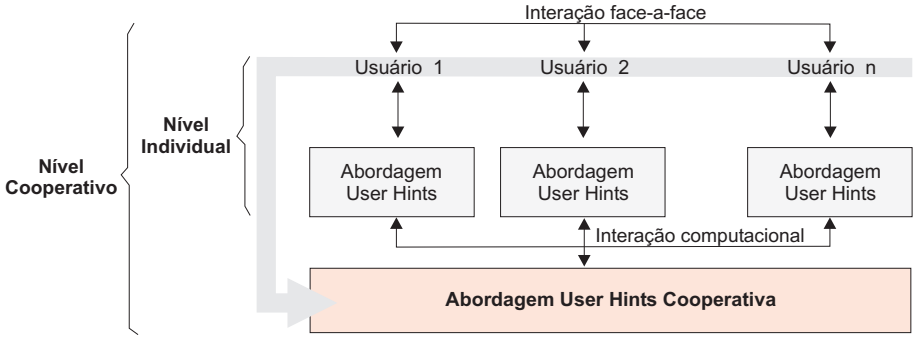


Figura 5.1: *Níveis de interação da abordagem Co-UserHints no modelo coletivo.*

No nível individual, o processo de otimização é o mesmo da abordagem *User Hints* tradicional. Cada usuário possui uma cópia do problema e trabalha de forma isolada, sem interação interpessoal, auxiliando métodos automáticos de otimização a construírem uma solução melhor.

Já no nível cooperativo, os usuários podem discutir o problema de otimização entre si e integrar os seus resultados individuais de forma a produzir uma solução de melhor qualidade. Nesse nível, duas formas de interação são possíveis:

- **Interação Face-a-Face:** como os usuários estão co-localizados, essa interação se dá através de expressões verbais e não-verbais, que, em geral, não envolvem diretamente recursos computacionais. Atividades visuais como indicação com as mãos e expressões da face podem ser utilizados para transmitir informações. Apesar de não exigir participação ativa de ferramentas computacionais, as aplicações de computador utilizadas nesse processo podem facilitar a comunicação entre os seres humanos [Scott, Carpendale e Inkpen 2004, Scott, Grant e Mandryk 2003], adotando, por exemplo, uma visualização clara do problema e das soluções que estão sendo discutidas, de modo a ampliar a consciência do espaço de trabalho (ver Seção 4.1.4 para maiores informações sobre consciência do espaço de trabalho).
- **Interação Computacional:** essa forma de interação é caracterizada pela utilização de recursos computacionais para integrar e compartilhar as soluções individuais geradas pelos usuários. As soluções podem ser integradas automaticamente pelo sistema ou quando solicitada por um dos participantes.

Uma descrição de como o processo de otimização ocorre e de todos os elementos da abordagem *Co-UserHints* nesse modelo são apresentados nas próximas seções.

5.3.1 Elementos da Abordagem

A Figura 5.2 ilustra a estrutura da abordagem *Co-UserHints* no modelo coletivo. A estrutura é composta dos seguintes elementos:

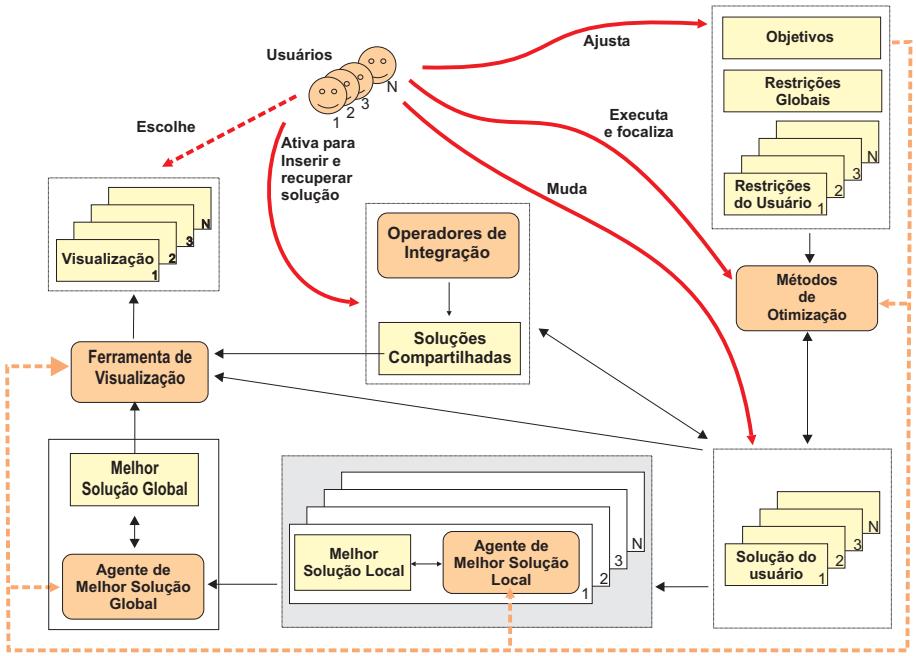


Figura 5.2: *Estrutura da abordagem Co-UserHints no modelo coletivo.*

- **Usuários:** são os operadores humanos que trabalham cooperativamente para obter a melhor solução para um problema de otimização;
- **Objetivos:** é uma função que define as características qualitativas da solução, de forma similar ao que é definido na abordagem *User-Hints*;
- **Restrições globais:** são restrições para as variáveis de todas as soluções de trabalho presentes no processo de otimização;
- **Restrições do usuário:** são restrições locais criadas pelos usuários unicamente para a sua solução de trabalho. Elas podem completar ou sobrepor as restrições globais;
- **Função de qualidade global:** computa a qualidade das soluções produzidas, de forma semelhante à função de qualidade descrita na Seção 3.2. Essa função considera, para fins de cálculo, a função objetivo e custos para não-satisfação de restrições globais do sistema;
- **Função de qualidade local do usuário:** inclui o valor da função objetivo e um custo para a não-satisfação de algumas restrições globais e locais do usuário. As funções de qualidade (local ou global) não estão represenadas na

Figura 5.2, contudo, as informações necessárias à qualidade das soluções estão indicadas por uma linha pontilhada;

- **Métodos de otimização:** são algoritmos utilizados pelos usuários durante o processo de otimização para produzir novas soluções;
- **Solução do usuário:** é a solução de trabalho corrente dos usuários. Cada usuário possui a sua solução de trabalho e pode alterá-la, através das dicas, para que essa alcance uma qualidade melhor;
- **Melhor solução local do usuário:** é a melhor solução criada pelo usuário durante o processo de otimização;
- **Melhor solução global:** é a melhor solução entre todas as soluções geradas por todos os usuários durante o processo de otimização;
- **Soluções compartilhadas:** são partes da soluções de trabalho ou soluções completas armazenadas pelos usuários em um local temporário, acessível a todos os participantes do processo de otimização;
- **Operadores de integração:** permitem unir duas ou mais soluções completas ou parciais, através de algoritmos e parâmetros definidos de acordo com o problema de otimização a ser resolvido. Esses operadores podem ser vistos como funções que recebem soluções como argumentos e retornam uma nova solução. Os operadores podem funcionar de modo automático ou serem ativados individualmente por cada usuário;
- **Agente de melhor solução local:** é um agente local para cada usuário que verifica automaticamente a qualidade da solução de trabalho e a grava como a melhor solução local, se esta tiver uma qualidade superior à melhor solução local corrente;
- **Agente de melhor solução global:** verifica automaticamente a qualidade de todas as melhores soluções locais, identificando entre elas a melhor solução global;
- **Ferramenta de visualização:** gera visualizações sobre o processo para os usuários;
- **Visualização:** podem representar as soluções particulares de cada usuário, bem como visões globais de todo o processo de otimização.

Neste modelo, cada usuário i, $i = 1, 2, ..., n$, está associado unicamente a um conjunto de restrições locais, a uma solução de trabalho, a um agente de melhor solução local, a uma função de qualidade local e a uma visualização particular.

5.3.2 Dicas Cooperativas

As dicas definidas na abordagem *User Hints* também valem na abordagem *Co-UserHints* (nos modelos coletivo e integrado). Contudo, novos tipos de dicas são definidos para os processos de otimização resolvidos cooperativamente, denominados *dicas cooperativas*. É principalmente por meio desses novos tipos de dicas que acontece a interação computacional entre os múltiplos usuários.

Três tipos de dicas cooperativas são exploradas:

- **Compartilhamento de Soluções**: durante o processo de otimização os usuários podem copiar parte da sua solução de trabalho ou toda ela para uma área compartilhada do sistema. As soluções armazenadas nessa área são visualizadas e acessadas por todos os membros do grupo. Esse tipo de dica permite, por exemplo, que os usuários dividam tarefas e compartilhem os resultados parciais ou completos obtidos individualmente.
- **Integração de Soluções**: operadores de integração podem ser projetados para integrar (combinar) soluções na área compartilhada ou soluções de trabalho dos usuários com essas últimas, visando obter um resultado de melhor qualidade. Esse resultado pode ficar armazenado na área compartilhada e ser acessível aos usuários da mesma forma como é feito no compartilhameto de soluções. Um operador de integração pode ser utilizado inclusive para recuperar soluções compartilhadas quando ativado por um usuário, ele substitui parte da solução de trabalho do mesmo pela solução compartilhada escolhida
- **Acesso à Solução Global**: cabe ao agente de solução global encontrar, entre todas as soluções individuais do grupo, uma única que seja a melhor. Essa verificação é feita automaticamente pelo agente e sem a ação dos usuários. Contudo, cada usuário pode acessar a melhor solução global armazenada, substituindo a sua solução de trabalho pela mesma bem como gravar a sua solução de trabalho como a melhor solução da cooperação.

Cada usuário pode usar as dicas já definidas pela abordagem *User Hints* e/ou as dicas cooperativas visando obter uma solução de melhor qualidade.

Convém destacar que a recuperação da solução anterior não foi explorada na abordagem *User Hints*. Contudo, Nascimento [Nascimento 2003] afirma que a possibilidade do usuário retornar para uma solução anterior é algo importante dentro de um processo de otimização. Portanto, neste trabalho a **recuperação da solução anterior** é considerado um tipo de "dica básica" (não cooperativa) disponível ao usuário através da nova abordagem *Co-User Hints*.

5.3.3 Processo de Otimização

Como no *User Hints Framework* o processo de otimização na abordagem *Co-UserHints* começa com a criação de uma solução inicial que é replicada para todos os usuários. Cada usuário é responsável pela otimização de sua solução de trabalho através das dicas e da execução dos métodos de otimização. Os agentes locais e global guardam, respectivamente, as melhores soluções locais e global durante todo o processo.

Através das dicas básicas (não cooperativas), um usuário modifica apenas a sua solução de trabalho, fazendo mudanças manuais e executando algoritmos, por exemplo. Nenhum usuário pode modificar a solução de outro diretamente, somente através das dicas cooperativas é possível realizar a troca e intregração de soluções entre os usuários.

Qualquer mudança que ocorra em uma solução de trabalho ativa o agente de melhor solução correspondente. Esse agente então verifica se a solução de trabalho possui uma qualidade superior a da melhor solução já criada pelo usuário, em caso afirmativo, a melhor solução é atualizada e o agente de melhor solução global é disparado. O agente local de melhor solução utiliza a função de qualidade local.

O agente local é notificado nas seguintes situações:

- o usuário executou alguma mudança manual na solução de trabalho;
- o usuário recuperou alguma solução compartilhada;
- o usuário recuperou alguma solução integrada;
- por ação de modificações ocorridas através do método de otimização;
- quando o usuário deseja gravar a sua solução de trabalho como a melhor solução local;
- quando o usuário deseja recuperar a melhor solução local como a sua solução de trabalho.

O agente de melhor solução global trabalha de forma similar. Ele compara todas as melhores soluções de cada usuário presentes na cooperação com a melhor solução global armazenada, atualizando-a. Assim como na abordagem *User Hints*, todos os agentes consideram a função de qualidade e as restrições para determinar as melhores soluções. O agente global de melhor solução utiliza a função de qualidade global para computar a qualidade da solução.

O agente global é disparado, na maioria das vezes, por ação dos agentes locais. Além disso, cada usuário da cooperação pode recuperar a solução global e, também, gravar a sua solução corrente como melhor global. Essa solução permanecerá como a melhor solução da cooperação até que ocorra uma modificação em

alguma melhor solução local ou até que haja mais uma intervenção direta do usuário. Sendo a solução global considerada como uma informação pública, a sua substituição manual envolve a consistência do sistema sendo necessário então que todos os usuários envolvidos no processo de otimização deêm o aceito para essa ação.

Os usuários também podem, a qualquer momento da otimização, utilizar os recursos de compartilhamento. Subsoluções podem ser selecionadas e armazenadas em uma área comum a todos. Qualquer usuário pode selecionar uma solução compartilhada e utilizá-la em sua própria solução de trabalho.

O processo de otimização é executado durante algum tempo com interações entre vários usuários e/ou interações entre os usuários e os métodos de otimização. No final, a solução armazenada pelo agente global é considerada a melhor solução para o problema. Todo o processo pode ser repetido se os usuários não estiverem satisfeitos com o resultado.

5.3.4 Resolução de Conflitos entre Restrições

Durante o processo de otimização na abordagem *Co-UserHints*, os usuários podem inserir restrições globais e locais que determinam características do problema que não são conhecidas. Essas restrições têm efeito na solução de cada usuário e são utilizadas tanto pelos métodos de otimização quanto pelos agentes de melhor solução para avaliar e comparar resultados.

O ajuste dinâmico de restrições globais e locais pelo usuário pode, contudo, criar situações de conflito para as quais não há solução viável. Por exemplo, em problemas de desenho de grafos, os usuários podem inserir uma restrição global que força determinado vértice a ficar sempre acima de outro, e uma restrição local conflitante defina exatamente o contrário, por exemplo, que o mesmo vértice deve ficar sempre abaixo de outro.

A forma de resolução de tais conflitos depende do problema a ser otimizado e da intenção dos usuários em relação a maneira de conduzir o processo de otimização. Tendo em consideração como se dará a convergência para a solução ótima pode-se priorizar as restrições globais ou as locais.

Na primeira possibilidade nenhum usuário pode inserir restrições locais que conflitam com as restrições globais já estabelecidas. Todos os usuários trabalham assim em busca de uma solução que convirja para um padrão estabelecido pelos objetivos e restrições globais do problema.

No caso de se priorizar restrições locais, essas podem ser inseridas livremente no problema, mesmo que conflitem com restrições globais. Havendo conflito, as restrições locais sobrepõem as globais. Este tipo de método pode ser utilizado para

permitir que os usuários busquem soluções alternativas mesmo que não estejam de acordo com as características gerais do problema.

Neste trabalho não foram utilizadas as mudanças manuais de restrições, desta forma, não foram executados testes para verificar o impacto destes conflitos no processo de otimização. Também, por não ser o objetivo desta pesquisa, este assunto não foi explorado em sua totalidade.

5.4 *Co-UserHints* no Modelo Integrado

No modelo integrado da abordagem *Co-UserHints*, todos os usuários têm acesso a uma única solução de trabalho compartilhada. Não existe assim o nível individual (ver Seção 5.3) apenas o cooperativo. Esse modelo pode ser visto como uma simplificação do anterior, onde há uma única visualização, um único agente de melhor solução (global), uma única função de qualidade e uma única solução de trabalho.

A Figura *Co-UserHints* 5.3 ilustra a estrutura geral da abordagem *Co-UserHints* no modelo integrado.

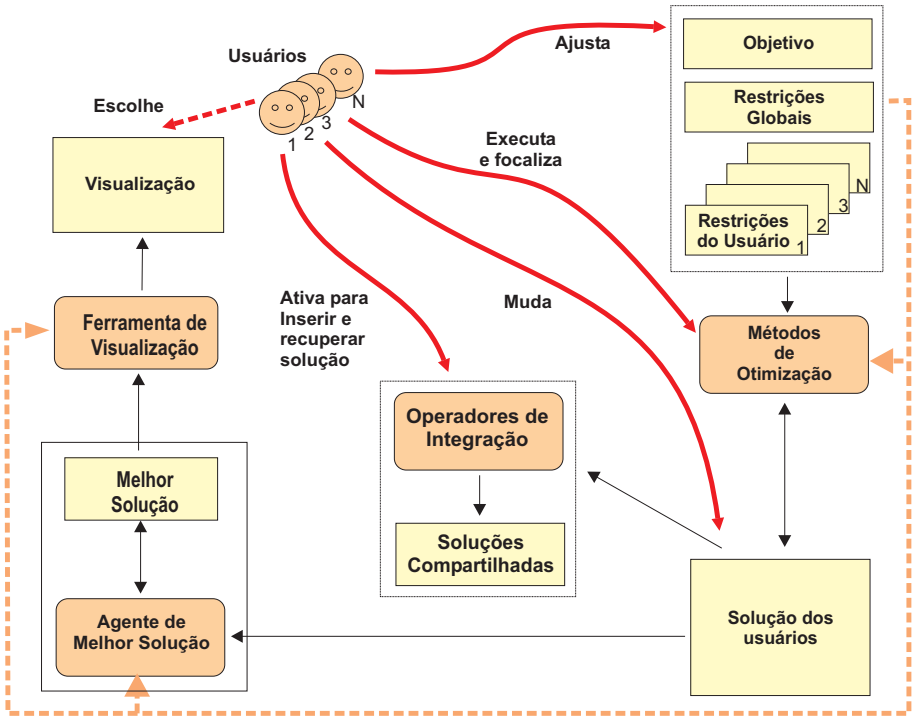


Figura 5.3: *Estrutura da abordagem Co-UserHints no modelo integrado.*

Comparado com o modelo coletivo, que possui agentes locais e global, o modelo integrado dispensa a presença de agentes locais de melhor solução, uma vez que, existe apenas uma solução de trabalho. Da mesma forma, a divisão entre

as funções de qualidade global e local deixa de existir, sendo utilizada agora somente uma função de qualidade global com mais parâmetros. Essa função leva em consideração o objetivo, as restrições globais e, as restrições de cada usuário, se existirem.

As restrições se mantêm como no modelo de integração, cada usuário pode impor restrições individualizadas, mas agora elas afetam o trabalho como um todo. Contudo, as restrições de caráter individual só podem ser alteradas pelo usuário que as criou, enquanto, as restrições globais podem ser ajustadas por qualquer usuário.

Neste modelo também podem acontecer conflitos de restrições entre os usuários e entre as restrições locais e globais. Esse problema pode ser resolvido de maneira mais simples de três formas automáticas:

- Impor a proibição pelo sistema de criar restrições conflitantes;
- Definir níveis de prioridade para as restrições conflitantes;
- Permitir restrições conflitantes deixando a cargo do sistema decidir qual restrição seguir para alcançar a melhor solução.

Como no modelo coletivo os conflitos de restrições não foram alvo desta pesquisa. Desta forma, não foram investigados de maneira mais produnda. Acredita-se que outras soluções são possívies sendo necessário um estudo mais completo.

Os demais elementos da abordagem preservam as mesmas funcionalidades definidas no modelo anterior.

5.4.1 Processo de Otimização

O processo de otimização é bastante parecido ao do modelo coletivo. Ele começa com a criação de uma única solução inicial para todos os usuários. Cada usuário através, das dicas e execução dos métodos de otimização, pode modificar essa solução conjuntamente. O agente de melhor solução de trabalho guarda uma cópia da solução inicial e a atualiza toda vez que uma solução de melhor qualidade é obtida.

Durante a otimização, os usuários podem interagir e tomar decisões sobre quais partes da solução corrente são colocadas na região de compartilhamento para serem recuperados depois ou integrados com alguma outra solução.

Para verificar os efeitos da cooperação de um grupo de pessoas co-localizadas em um problema de otimização de um desenho de grafo direcionado, utilizando a abordagem *Co-UserHints* (nos dois modelos), foram desenvolvidas duas aplicações multiusuário que serão descritas no próximo capítulo.

CAPÍTULO 6

Aplicações Co-GDHints para um Problema de Desenho de Grafos

Este capítulo apresenta duas aplicações interativas e cooperativas para a resolução de um problema de otimização de desenho de grafos direcionados, construídas utilizando a abordagem *Co-UserHints*.

6.1 O Problema de Desenho de Grafos Direcionados

A área de desenho de grafos envolve diversos problemas complexos de otimização [Battista et al. 1999], dentre os quais destaca-se a geração de desenhos de grafos direcionados. Como nesse tipo de grafo as arestas possuem um sentido, ele é usado, via de regra, para representar estruturas hierárquicas como, por exemplo, autômatos e diagramas de fluxo de dados.

A maioria dos algoritmos tradicionais para desenho de grafos direcionados procura construir desenhos que enfatizam tal hierarquia de informações, ao mesmo tempo que atendem a outros critérios estéticos. Esses algoritmos normalmente utilizam o Paradigma Hierárquico descrito na Seção 2.2.2 e satisfazem aos critérios estéticos na seguinte ordem de prioridade (do mais prioritário ao de menos importância):

- apresentar arestas com orientação uniforme (por exemplo, voltadas para cima ou para baixo);
- mostrar poucos cruzamentos de arestas;
- evitar desenhar arestas muito longas;
- reduzir a área total do desenho.

Esta dissertação investiga o trabalho cooperativo em desenho de grafos direcionados. Contudo, é considerada uma variação do problema de desenho de grafos direcionados tradicional. A minimização de cruzamentos de arestas é o critério de

maior prioridade, as arestas são representadas por linhas retas[1] e não se trata a padronização do comprimento das arestas. Isso se deve a grande complexidade de implementação de algoritmos para construção de desenhos direcionados no modelo tradicional, o que prolongaria por demais a realização deste trabalho.

Desta forma, o padrão gráfico adotado aqui consiste em representar os vértices como pequenos retângulos rotulados e as arestas como linhas retas. Sendo o desenho realizado em um plano cartesiano como ilustrado na Figura 6.1.

A meta do processo de desenho é satisfazer a três critérios estéticos na seguinte ordem de prioridade:

- minimização do número de cruzamentos entre arestas;
- padronização da orientação das arestas e;
- minimização da área total do desenho.

É importante destacar que a otimização de todos esses critérios não é fácil de ser alcançada uma vez que envolve a resolução de problemas NP-difíceis (como descrito na Seção 2.2.6). Os critérios também são conflitantes entre si, pois na maioria das vezes a melhora de um implica na piora dos demais.

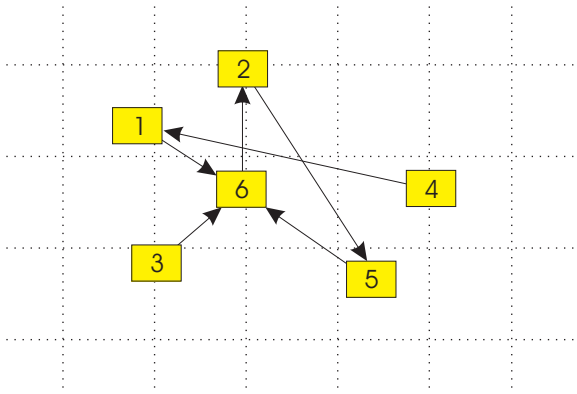


Figura 6.1: *Padrão de representação de desenhos de grafos.*

Foram desenvolvidas duas aplicações multiusuário denominadas *cCo-GDHints* e *iCo-GDHints* para o problema de desenho de grafos direcionados. A primeira aplicação implementa o modelo coletivo da abordagem *Co-UserHints* e a segunda, o modelo integrado, conforme discutidos na Seção 5.1.

O *cCo-GDHints* divide o espaço de trabalho em áreas privadas com cada usuário trabalhando em uma região distinta. Essas áreas possuem visões individuais do desenho de grafo, isto é, cada usuário possui em sua área de edição uma

[1]No modelo tradicional de desenho de grafos direcionados as arestas são modeladas por segmentos de retas, podendo apresentar curvas.

visualização do desenho de grafo, que é a sua solução de trabalho particular. Esses desenhos são depois integrados pelo sistema e/ou pelos usuários cooperativamente. Já na aplicação *iCo-GDHints*, a visão é única e só existe uma área de edição onde todos usuários trabalham conjuntamente.

Na próxima seção são apresentadas as características comuns às duas aplicações.

6.2 Características Comuns das Aplicações

Os sistemas *cCo-GDHints* e *iCo-GDHints* foram projetados como aplicações SDG para serem usados simultaneamente por duas pessoas co-localizadas. Eles utilizam o pacote *SDGToolkit* (ver Seção 4.1.6), desenvolvido por Tse [Tse e Greenberg 2002] para gerenciar dois dispositivos de *mouse* ao mesmo tempo.

O *cCo-GDHints* e o *iCo-GDHints* implementam quase todos os elementos da nova abordagem *Co-UserHints*, com exceção da criação e modificação dinâmica de restrições. Sete tipos de dicas foram inseridas nesses sistemas:

- mudanças manuais no desenho através de seleção e movimentação dos vértices;
- focalização dos métodos de otimização;
- gravação de uma solução particular como a melhor solução local ou global;
- recuperação da melhor solução local ou global;
- inserção ou recuperação de uma solução completa ou parcial em uma área de compartilhamento;
- integração de duas soluções usando operadores de integração. Um operador criado especificamente para integrar dois desenhos de grafos será descrito na Seção 6.2.2;
- recuperação de uma solução anterior.

Durante o processo de otimização, os usuários podem utilizar essas dicas para melhorar um desenho inicial gerado de modo automático.

Não foi implementado, no entanto, a dica de criação e modificação dinâmica de restrições, pois, durante os testes da abordagem *User Hints* para problemas de desenho de grafos, Nascimento [Nascimento 2003] relatou ter encontrado dificuldades para projetar e avaliar sistemas efetivos que permitissem a modificação interativa e dinâmica de restrições. Ele verificou que é difícil gerar uma solução esteticamente agradável, que atenda as novas restrições impostas dinamicamente pelo usuário, sem perder o mapa mental do desenho. Além disso, ainda não há uma abordagem

clara para avaliar os resultados gerados por restrições especificadas subjetivamente pelos usuários. Por razões semelhantes, as aplicações *cCo-GDHints* e *iCo-GDHints* também não implementam mudanças dinâmicas da função objetivo.

6.2.1 Cálculo da Qualidade dos Desenhos

Como para esses sistemas não existem inserção e remoção de restrições dinâmicas, a função de qualidade e a função objetivo são idênticas e consistem de um simples vetor de qualidade $Q(D)$, o qual mede o custo de um desenho D para cada um dos critérios especificados.

O vetor de qualidade $Q(D)$ é definido como:

$$Q(D) = \langle q_1(D), q_2(D), q_3(D) \rangle \tag{6-1}$$

Onde $q_1(D)$ é a quantidade de cruzamentos entre as arestas do desenho, $q_2(D)$ é a quantidade de arestas voltadas para cima e $q_3(D)$ é a área total do desenho.

Os algoritmos de otimização, os agentes de melhor solução e os operadores de integração utilizam esses vetores para avaliar as soluções.

Toda solução da otimização (seja uma solução de trabalho ou uma melhor solução) possui associado um vetor de qualidade. O objetivo é minimizar $Q(D)$ em ordem de prioridade das posições de seus parâmetros, ou seja, primeiro $q_1(D)$, depois $q_2(D)$ e por último $q_3(D)$.

Um desenho D_a é dito *melhor* que um desenho D_b e representado por $D_a < D_b$ se,

$$q_1(D_a) < q_1(D_b) \tag{6-2}$$

ou

$$q_i(D_a) = q_i(D_b) \tag{6-3}$$

para $\forall K < i \leq 3$, com $K \in \{1, 2\}$ e $q_{i+1}(D_a) < q_{i+1}(D_b)$

Se D_a e D_b possuírem a mesma qualidade (independentes ou não de serem desenhos iguais) esse fato é representado como $D_a = D_b$.

6.2.2 Algoritmos de Otimização e de Integração

Os sistemas *cCo-GDHints* e *iCo-GDHints* utilizam os mesmos algoritmos de otimização. Dois métodos de otimização foram implementados: uma versão modificada do algoritmo *Spring Embedder* descrito na Seção 2.2.2 e uma simplificação do

algoritmo Davidson e Harel (ver Seção 2.2.2). Os dois algoritmos são métodos clássicos para grafos gerais adaptados ao problema de grafos direcionados; conseqüentemente, o critério de direção de arestas não é trabalhado de forma efetiva. Isse fato não afeta de modo negativo o cálculo da qualidade do desenho, uma vez que, o critério considerado de maior prioridade é o número de cruzamentos de arestas e, os dois algoritmos conseguem trabalhar de maneira efetiva esse parâmetro.

Além desses dois algoritmos, foi implementada também uma heurística construtiva gulosa, usada como um operador de integração, para unir dois ou mais desenhos de grafos. Esses três algoritmos são descritos a seguir.

Método de Otimização *Spring Embedder* Modificado

O algoritmo *Spring Embedder*, explicado na Seção 2.2.7, foi modificado para agir somente enquanto há melhoria na qualidade de um desenho de acordo com a função de qualidade $Q(D)$ definida na seção anterior. Essa modificação foi necessária, uma vez que, os métodos de força direcionada possuem, via de regra, uma ação maior na simetria[2] do desenho.

Assim, eles costumam minimizar a quantidade de cruzamentos no início, mas depois, para alcançar um ponto de energia mínima (simetria), podem encaminhar-se para uma solução que aumenta o número de cruzamentos. A estrutura do algoritmo modificado é mostrada abaixo, foi utilizada a mesma notação da Seção 2.2.7:

[2]Não é objetivo desse trabalho avaliar o critério estético "simetria", todavia, o algoritmo *Spring Embedder* naturalmente produz desenhos simétricos, assim, tal critério acaba estando indiretamente presente nos desenhos produzidos. Contudo, não é computada para efeito de cálculo da qualidade.

6.1: $SpringEmbedderModificado(G, D)$

Entrada: Desenho D de um grafo $G = (V, E)$.

Saída: Desenho D modificado.

```
1  D_t = D
2  contador = 1
3  enquanto contador > M faça
4  |  para cada u ∈ V faça
5  |  |  Calcular a força resultante f(u) das molas conectadas ao
   |  |  vértice u no desenho D_t
6  |  fim
7  |  para cada u ∈ V faça
8  |  |  p_u^{D_t} = p_u^{D_t} + c × f(u)
9  |  fim
10 |  se D_t < D então
11 |  |  D = D_t
12 |  fim
13 |  contador = contador + 1
14 fim
15 retorna D
```

Os termos c e M são constantes e D_t é um desenho de grafo temporário sobre o qual é executado o método de força direcionada.

O algoritmo começa calculando a força resultante em todos os vértices do desenho temporário D_t. Depois cada vértice é movido no sentido da força. Após essas iterações compara-se a qualidade final do novo desenho D_t com a qualidade do desenho D. Se a qualidade de D_t for melhor do que a de D então o desenho D_t é copiado em D. Esses passos se repetem M vezes.

Método de Otimização Davidson e Harel Modificado

O método original Davidson e Harel, descrito na Seção 2.2.7, é um *simulated annealing*. Esse método entretanto foi modificado na presente pesquisa para trabalhar com a função de qualidade $Q(D)$ e para funcionar como um algoritmo guloso.

6.2: $DavidsonHarelModificado(G, D)$

Entrada: Desenho D de um grafo $G = (V, E)$.
Saída: Desenho D modificado.

```
1  Determinar um raio inicial r = r_0
2  D_t = D
3  contador = 1
4  enquanto contador > M faça
5  |  Escolhe um vértice aleatório u de V
6  |  p_ant = p_u^{D_t}
7  |  p_u^{D_t} = p_u^{D_t} + δ
8  |  se D < D_t ou D = D_t então
9  |  |  p_u^{D_t} = p_ant
10 |  senão
11 |  |  D = D_t
12 |  fim
13 |  r = r × 0.99
14 |  contador = contador + 1
15 fim
16 retorna D
```

O parâmetro δ é um vetor que determina quanto e em qual direção mover um vértice. Este vetor é calculado aleatoriamente dentro de um raio r. M é uma constante e D_t é um desenho de grafo temporário sobre o qual é executado o método.

O algoritmo começa escolhendo um vértice aleatório do desenho D_t. Esse vértice então é movido para uma outra posição conforme δ. Testa-se a qualidade final do novo desenho D_t comparando a sua qualidade com a do desenho de trabalho D. Se a qualidade de D_t for melhor do que a qualidade de D então a posição dos vértices nos dois desenhos são igualadas; caso contrário, o vértice escolhido de D_t volta para a posição anterior. Esses passos se repetem M vezes.

Método de Integração de Desenhos

Para investigar a utilização de operadores de integração, foi implementado um algoritmo baseado em heurísticas construtivas gulosas. Uma heurística construtiva produz soluções construindo elemento por elemento. O termo guloso indica que o elemento a ser inserido a cada passo é aquele que apresenta melhor valor imediato segundo o critério adotado.

O algoritmo desenvolvido é mostrado abaixo:

6.3: $MetodoGuloso(G_a, G_b, D_a, D_b)$

Entrada: Desenhos D_a de um grafo $G_a = (V_a, E_a)$ e D_b de um grafo $G_b = (V_b, E_b)$

Saída: Desenho D_a modificado com base nas informações de D_b.

```
1  V_a = V_a ∪ V_b
2  D_t = D_a
3  Determinar um contador inicial M =| D_a |
4  i = 1
5  enquanto contador > M faça
6  |  Escolher o i-ésimo vértice u de V_b;
7  |  p_ant = p_u^{D_t}
8  |  p_u^{D_t} = p_u^{D_b}
9  |  se D_t < D_a então
10 |  |  D_a = D_t
11 |  |  i = 1
12 |  senão
13 |  |  p_u^{D_t} = p_ant
14 |  fim
15 |  i = i + 1
16 fim
17 retorna D_a
```

No algoritmo os termos D_a e D_b são, respectivamente, desenhos dos subgrafos $G_a = (V_a, E_a)$ e $G_b = (V_b, E_b)$ de $G = (V, E)$. D_t representa um desenho de grafo temporário sobre o qual é executado o método.

O método de integração considera um desenho de grafo inicial D_a e integra um outro desenho, denominado D_b, a este. A solução é construída da seguinte forma: o algoritmo começa adicionando à V_a os vértices de D_b que não estão em D_a. Depois, escolhe-se o i-ésimo vértice u de D_b e move o vértice correspondente a este em D_t (D_t é cópia de D_a) para a mesma posição de u. Se a qualidade de D_t for melhor do que a qualidade de D_a então a posição dos vértices nos dois desenhos são igualadas e o contador i é reinicializado. Caso contrário, o vértice escolhido de D_t volta para a posição anterior.

Sempre que a posição de um vértice u em D_t é modificada o processo é reiniciado, pois a nova posição de u pode fazer com que movimentos de outros vértices rejeitados anteriormente sejam aceitos agora. O método só termina quando todos os vértices de D_b forem analisados e nenhuma melhoria tiver sido identificada.

O algoritmo guloso de integração acima é utilizado no sistema *cCo-GDHints* tanto pelos operadores manuais de integração quanto pelo agente automático de melhor solução integrada, os quais são descritos na Seção 6.3. A sua grande vantagem é proporcionar uma abordagem mais efetiva para combinar dois desenhos que apresentam coordenadas distintas para um mesmo conjunto de vértices.

6.3 *GDHints* Cooperativo Coletivo

O sistema *GDHints* Cooperativo Coletivo (*Collective Cooperative GDHints - cCo-GDHints*) é um sistema SDG que implementa o modelo coletivo da abordagem *Co-UserHints* permitindo que um grupo de pessoas trabalhem em desenhos de grafo simultaneamente. Esse sistema foi inicialmente implementado com suporte a quatro usuários, sendo três locais e um remoto [Ferreira, Nascimento e Albuquerque 2006, Ferreira e Nascimento 2005]. Contudo, devido a dificuldades de realizar experimentos controlados nesse ambiente decidiu-se por simplificar a *interface* para apenas dois participantes.

Com exceção aos objetivos e restrições dinâmicas, o sistema *cCo-GDHints* implementa todos os elementos do modelo coletivo. Em adição a esse modelo, ele contém ainda um novo agente, chamado agente de integração. Esse agente integra automaticamente os melhores desenhos dos usuários com um desenho de grafo já armazenado pelo mesmo. O resultado dessa integração automática é mostrada em uma área da *interface* sendo que qualquer usuário pode recuperar tal solução.

Foi implementado ainda um operador de integração que possui a mesma funcionalidade do agente de integração, com a diferença de que o operador é manual, ou seja, deve ser acionado por usuários.

O agente de integração automático e o operador de integração utilizam o algoritmo de integração descrito na Seção 6.2.2. Note que esse algoritmo garante que a solução integrada terá qualidade maior ou, no mínimo, igual a qualidade da melhor solução integrada corrente.

6.3.1 *Interface* da Aplicação

A *interface* da aplicação *cCo-GDHints* é dividida em cinco regiões, como apresentado na Figura 6.2 e descrito abaixo.

- duas áreas de edição idênticas e individuais, uma para cada usuário, que estão localizadas na parte inferior da tela e essas áreas possuem uma barra de ferramentas com opções para executar os métodos de otimização, recuperar a melhor solução local, global ou integrada, e ferramentas de edição como

fazer/desfazer, *zoom* e "selecionar tudo". Nenhum usuário tem permissão para editar a área de outro participante;

- uma área de visualização da melhor solução global, localizada no canto superior esquerdo. Essa área é atualizada toda vez que a melhor solução global é modificada, e não pode ser alterada pelos usuários;
- uma região que mostra o melhor desenho produzido pelo agente de integração automático, localizada no canto superior direito e essa área é atualizada sempre que esse agente consegue produzir um desenho melhor do que a melhor solução integrada. Nenhum usuário pode editar diretamente o desenho exposto nesse espaço;
- a área de compartilhamento no centro da tela, no lado esquerdo, serve como um espaço de armazenamento temporário de soluções. Todos os usuários têm acesso às soluções armazenadas nessa região. Contudo, não podem alterar de forma direta as soluções lá armazenadas, ou seja, a modificação de qualquer solução do espaço compartilhado significa ter que recuperá-la para a área de edição, alterar e depois compartilhá-la novamente;
- uma área para integração manual de soluções no centro da tela, no lado direito e cada usuário pode usá-la para integrar suas soluções manualmente com uma solução pré-gravada, através do acionamento de um operador de integração manual. Se esta área estiver vazia a solução selecionada pelo usuário é armazenada sem alteração. Na próxima ação de integração, a nova seleção do usuário é integrada com a solução anterior que ficou armazenada. Essa é uma região pública que pode ser utilizada por todos os usuários.

O sistema requer como entrada um arquivo que contém as arestas e vértices de um grafo. Um desenho desse grafo é então mostrado na tela e os usuários podem interagir com o sistema através das dicas implementadas. Como o grupo está co-localizado a interação face-a-face pode ocorrer.

Também são solicitados no início da aplicação os nomes dos dois usuários. Essa informação é importante para rotular os ponteiros dos *mouse* e para identificar o autor de algumas ações. Cada usuário é identificado no sistema pelo seu nome.

A área de compartilhamento serve como espaço de armazenamento temporário onde cada usuário pode colocar uma solução parcial ou completa para ser recuperada em outro momento. Para salvar uma solução na área de compartilhamento um usuário deve selecionar uma região (vértices) no seu desenho de trabalho (ou todos eles) e arrastá-la para a área de compartilhamento. Um ícone é criado indicando que a solução foi armazenada. A área de compartilhamento tem um limite de dez soluções.

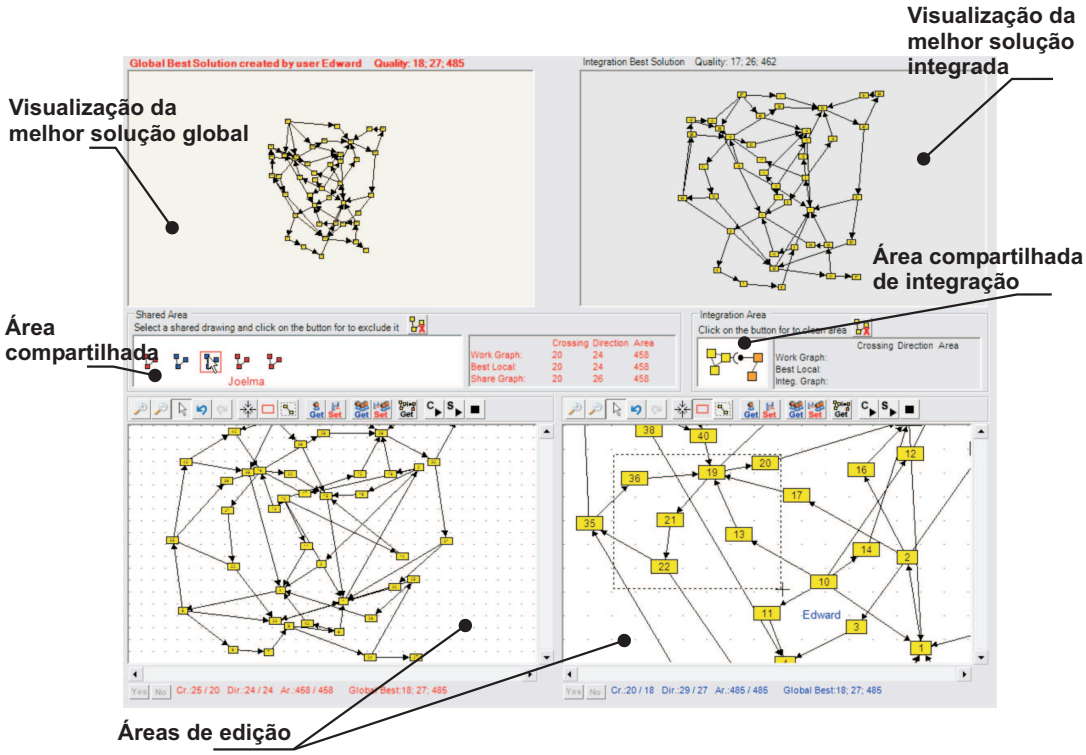


Figura 6.2: *Interface gráfica da aplicação cCo-GDHints.*

A recuperação é semelhante: um usuário seleciona algum ícone da área de compartilhamento e arrasta-o para a sua área de edição. Nesse momento, a solução de trabalho é alterada com base no desenho trazido da área compartilhada. Essa alteração é realizada através de um algoritmo que simplesmente substitui as coordenadas do vértice na área de trabalho pelas coordenadas da seleção compartilhada.

No processo de integração de uma solução compartilhada com a solução de trabalho, usou-se a idéia de "ponto médio"[3]. Os vértices do desenho compartilhado possuem uma posição (x, y) herdada do desenho original. Contudo, quando esse desenho é substituído no desenho de trabalho os vértices não são colocados em suas posições absolutas originais mas, antes os ponto médio do desenho compartilhado, e da mesma região no desenho de trabalho são calculados e as posições dos vértices do desenho compartilhado são atualizadas tendo como eixo os dois ponto médios igualados. Isso é importante para que a substituição não altere de forma brusca o mapa mental do usuário em relação a sua solução de trabalho.

Note que a área de compartilhamento permite combinar partes de soluções de usuários distintos ou a solução corrente de um usuário com parte de uma solução produzida anteriormente pelo mesmo. No entanto, o operador manual de

[3]O ponto médio é calculado através da média aritmética entre a posição (x, y) do vértice mais a esquerda e superior com a posição (x, y) do vértice mais a direita e inferior da seleção.

integração disponível na *interface* permite uma forma mais inteligente de combinação de soluções.

6.3.2 Processo de Otimização

A otimização começa com a escolha do grafo de trabalho. Em seguida o sistema gera um desenho inicial onde os vértices são colocados em posições aleatórias do espaço de visualização.

Essa solução inicial é replicada para todos os usuários da cooperação. Cada usuário trabalha individualmente em sua área de edição, podendo selecionar e mover os vértices e ainda executar os algoritmos de otimização, conforme definido na abordagem *Co-UserHints*. Uma cópia do desenho de grafo inicial é armazenada também pelo agente de melhor solução global e pelos agentes locais de cada usuário. O agente de integração automático utiliza ainda essa solução inicial como a melhor solução integrada inicial.

A avaliação da qualidade da solução de trabalho de cada usuário é feita automaticamente pelo sistema. Os agentes locais checam a qualidade do desenho de trabalho do usuário, localizado na área de edição, e armazenam o melhor resultado gerado. Em seguida acionam o agente global e o agente de integração. O agente global atualiza a melhor solução global se for pertinente e o agente de integração tenta integrar a melhor solução fornecida pelo agente local com a melhor solução integrada.

A qualquer momento, os usuários podem recuperar o seu melhor desenho local, o melhor desenho global produzido pelo grupo ou ainda a solução armazenada pelo agente automático de integração.

Como cada usuário tem uma área particular do desenho, as soluções dos mesmos podem ser diferentes. Através da área de compartilhamento, esses usuários podem compartilhar partes de suas soluções ou os seus desenhos completos. Isso possibilita, entre outras estratégias de cooperação, uma divisão de tarefas: os usuários podem delimitar a área de atuação de cada um no desenho e, utilizar recursos de compartilhamento posteriormente para agregar pedaços de soluções. Entretanto, estas estratégias não são obrigatórias e nem suportadas de forma explícita pelo sistema, são, na verdade, dependentes de uma estratégia de trabalho do grupo.

Cada usuário pode usar ainda a integração manual de parte ou todo o desenho, através do acionamento do operador de integração manual. Essa integração pode acontecer de usuário para usuário ou somente com soluções alternativas de um único usuário.

Algumas ações manuais no sistema afetam o trabalho de todos os usuários e precisam, portanto, de uma aprovação prévia dos mesmos. São exemplos dessas ações: eliminar soluções compartilhadas e gravar a solução corrente do usuário como melhor solução global.

Um outra ação que precisa ser permitida por todos os usuários é a gravação de uma solução de trabalho como a melhor solução global.

6.3.3 Consciência do Espaço de Trabalho

Para auxiliar a mantenção da consciência do usuário sobre o espaço de trabalho foram inseridos padronizações de cores, som, informações textuais e gráficas.

Cada usuário é identificado por uma cor (vermelha ou azul), cores essas que são utilizadas para diferenciar os rótulos do cursor dos dispositivos de *mouse*, com o nome do usuário, e as informações textuais escritas na área de edição de cada usuário. Essas informações textuais indicam a qualidade da solução de trabalho corrente, da melhor solução local e da melhor solução global.

Uma outra diferenciação através de cores é com respeito a área de compartilhamento. Sempre que um usuário insere um desenho na área de compartilhamento o sistema gera um ícone identificando esse usuário (através da cor). Quando esse ícone é selecionado por outro usuário, ele mantém a cor original de quem o criou, mas a cor da seleção é igual a do usuário que está selecionando tal ícone.

Além desse alteração na imagem do ícone, informações sobre como ficará a qualidade do desenho de trabalho caso o usuário incorpore à este a solução compartilhada são apresentadas na tela. A seleção do ícone da região integrada mostra o mesmo tipo de informação, sempre na coloracão padrão do usuário.

Para auxiliar a percepção de um contexto comum de trabalho, uma imagem da melhor solução global e o nome do usuário que a gerou são apresentadas em uma região do espaço de trabalho. Sempre que a melhor solução global é atualizada sua imagem na tela é alterada e um sinal sonoro é emitido. Cada usuário está associado a um som diferente. Dessa forma, eles podem identificar quem gerou a melhor solução global sem precisar procurar esta informação explicitamente na tela.

A imagem e os dados sobre a qualidade da melhor solução integrada também são apresentados na tela e atualizados toda vez que a solução integrada muda.

6.3.4 Vantagens e Desvantagens

O sistema *cCo-GDHints* tem como vantagem auxiliar as atividades individuais de cada usuário. Como cada pessoa possui a sua área de edição particular ela

pode montar estratégias individuais para a resolução do problema sem interferir no trabalho do colega.

A principal desvantagem do sistema *cCo-GDHints*, por sua vez, é a dificuldade em se manter o contexto compartilhado. A consciência sobre o que o outro está fazendo e, por conseqüência, a percepção do processo de otimização como um todo ficam prejudicados devido à individualização do problema.

Além disso, essa individualização pode causar competitividade entre os colaboradores, o que pode até pode ser benéfica se incitar nos participantes o desejo de melhora da solução.

Os conflitos de interesses podem não atrapalhar a otimização diretamente, mas o fato dos usuários seguirem caminhos controversos durante o processo pode fazer com eles não cheguem a uma boa solução.

É necessário, portanto, que estratégias coletivas de trabalho sejam muito bem definidas e utilizadas, e de preferência no início do processo de otimização.

6.4 *GDHints* Cooperativo Integrado

O sistema *GDHints* Cooperativo Integrado (*Integrated Cooperative GDHints iCo-GDHints*) também é um sistema SDG para desenho de grafos direcionados para um grupo de duas pessoas co-localizadas. Foi implementado utilizando o modelo integrado da abordagem *Co-UserHints*.

6.4.1 *Interface* da Aplicação

Esse sistema é mais simples do que a aplicação *cCo-GDHints*. Sua *interface* está de acordo com o padrão WYSIWIS (ver Seção 4.1.5), ou seja, os usuários compartilham a mesma visão do desenho, diferente da metodologia coletiva discutida na seção anterior onde os usuários tem uma visão individual.

A *interface* do *iCo-GDHints* é dividida em duas regiões apenas, conforme mostrado na Figura 6.3.

- **uma área de compartilhamento de soluções no topo da tela:** essa é uma região pública que pode ser acessada por todos os usuários para armazenar soluções parciais ou completas;
- **uma área de edição pública onde todos os usuários trabalham simultaneamente no mesmo desenho:** essa área possui uma barra de ferramentas com opções para executar os métodos de otimização, recuperar a melhor solução global, e com ferramentas de edição como "fazer/desfazer" e selecionar tudo.

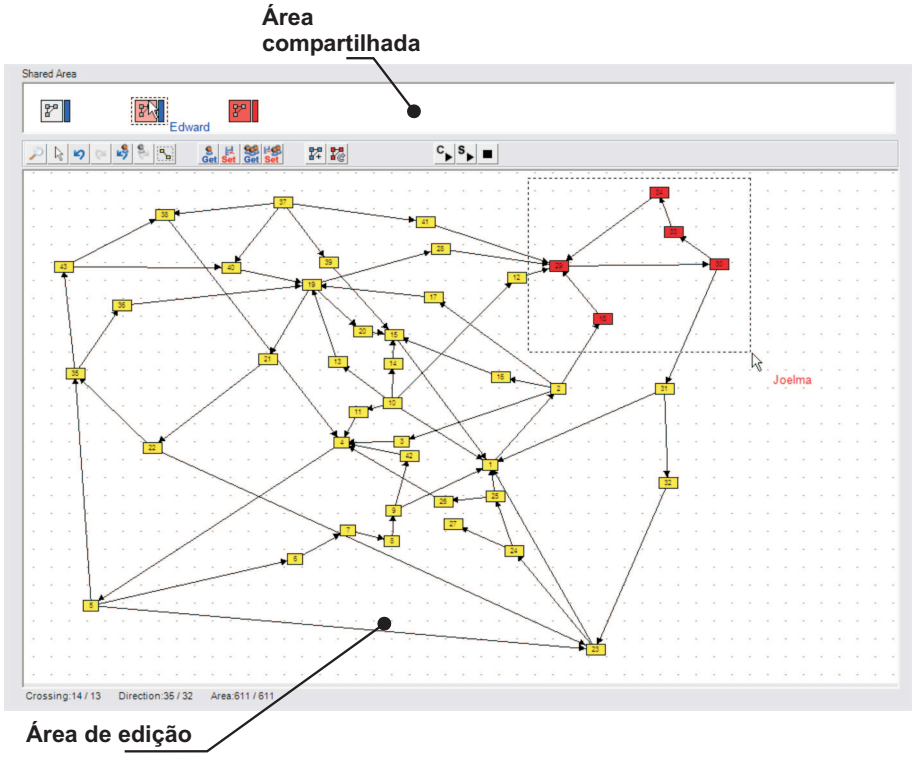


Figura 6.3: *Interface gráfica da aplicação iCo-GDHints.*

O sistema *iCo-GDHints* possui muito menos recursos de interação do que o sistema *cCo-GDHints*. Não foram implementados no mesmo a opção de "arrastar-e-soltar" múltiplos vértices selecionados.

Como conseqüência, a edição de um desenho deve ser feita movimentando-se um vértice por vez. Além disso, a troca de dados entre a área de trabalho e a área de compartilhamento é realizado através de botões na barra de ferramentas.

Também não foram implementados barras de rolagem e controles de *zoom*; esses dois recursos em especial teriam um impacto grande no sistema, pois poderiam interferir na visualização da informação por outros integrantes da cooperação. Todavia, foi implementado a opção de "desfazer/refazer" para múltiplos usuários, que permite a cada usuário desfazer a sua própria ação sem interferir na ação de outro.

Os algoritmos podem ter a sua ação focalizada em uma região do desenho, contudo, os usuários não podem executar dois algoritmos simultaneamente em duas regiões diferentes do desenho. Isso por que não está implementado a seleção simultânea em regiões distintas do desenho.

O *iCo-GDHints* também não possui operadores de integração, nem agente de integração automático. Dessa forma, esse sistema oferece apenas seis tipos de dicas:

- mudanças manuais no desenho através de seleção e movimentação dos vértices;

- focalização e execução de dois métodos de otimização;
- gravação de uma solução particular como a melhor solução global;
- recuperação da melhor solução global;
- inserir ou remover uma solução da área de compartilhamento; e
- recuperação da solução anterior, possuindo a característica, também, de ser uma recuperação multiusuário, ou seja, cada pessoa pode desfazer uma ação particular sua sem interferir nas ações executadas pelos os outros.

6.4.2 Processo de Otimização

A otimização começa da mesma maneira que ocorre no sistema *cCo-GDHints*: com a identificação dos usuários, a escolha do grafo de trabalho e a geração de uma solução inicial. Essa solução é repassada ao único agente existente que a considera como a melhor solução global. Os usuários começam então a editar a solução de trabalho ou a focalizar a execução dos algoritmos de desenho sobre regiões da mesma, visando obter um desenho de melhor qualidade.

A avaliação da qualidade da solução de trabalho é feita automaticamente pelo sistema. Toda vez que a solução de trabalho é alterada o agente de melhor solução é acionado para armazená-la. Qualquer usuário pode recuperar o melhor desenho produzido pelo grupo.

A área de compartilhamento não possui mais a função de área para troca de solução entre os usuários, servindo agora apenas como um ambiente de armazenamento temporário de soluções para os participantes.

6.4.3 Vantagens e Desvantagens

A principal vantagem do sistema *iCo-GDHints* é promover um forte senso de contexto compartilhado para os usuários. O fato de estarem compartilhando o mesmo desenho ajuda cada usuário a desenvolver uma consciência maior sobre o que o outro está fazendo e por conseqüência, perceber o processo de otimização como um todo.

A montagem de estratégias coletivas se torna mais natural nesse sistema, pois o padrão WYSIWIS traz para o processo de otimização a sensação de objetivo único para todos os usuários da colaboração.

Como desvantagem, destaca-se a falta de individualidade dos usuários. Cada um é forçado a trabalhar dentro de uma estratégia comum. Isso pode causar conflitos de interesses que precisam ser resolvidos pelos próprios usuários.

A coordenação das atividades também é um ponto dificultante. Como todos os usuários têm acesso a todas as funcionalidade do sistema, deve existir uma coordenação explícita das atividades. Por exemplo, se um usuário resolve executar um algoritmo sem consultar o colega, isso causa um interrupção na atividade deste e até a perda do trabalho.

Os dois sistemas descritos neste capítulo foram avaliados em estudos pilotos com o objetivo de identificar os padrões de trabalho cooperativo que mais se adequam ao problema de desenho de grafos. A metodologia utilizada nesses estudos e os resultados obtidos são apresentados no próximo capítulo.

CAPÍTULO 7

Avaliação do Trabalho

Neste capítulo é apresentada uma avaliação dos sistemas *cCo-GDHints* e *iCo-GDHints*. Experimentos foram realizados visando identificar se usuários trabalhando cooperativamente conseguem obter resultados melhores do que pessoas desenhando grafos sozinhas. Também foi verificado quais configurações de ambientes cooperativos podem ajudar na atividade de desenho de grafos. Os experimentos foram executados em dois estudos pilotos e os resultados são relatados no final do capítulo.

7.1 Primeiro Estudo Piloto

O primeiro estudo piloto teve como objetivo comparar os resultados de um grupo de usuários desenhando grafos isoladamente com pares de usuários trabalhando cooperativamente nos modelos coletivo e integrado da abordagem Co-UserHints.

7.1.1 Participantes

O estudo foi realizado com sessenta estudantes (homens e mulheres) voluntários, sendo cinquenta e seis do curso de graduação em Sistemas de Informação das Faculdades Alves Faria e quatro do curso de mestrado em Ciência da Computação da Universidade Federal de Goiás. Todos os participantes tinham familiaridade com computadores, *mouse* e o sistema operacional *Windows*. Não tinham, contudo, experiência com desenho de grafos e com aplicações *groupware*, com exceção dos quatro estudantes de mestrado que já haviam usado o sistema *cCo-GDHints* em testes anteriores.

7.1.2 Recursos Computacionais

Os experimentos foram feitos em estações Athlon de 64 *bits* com 512Mb de memória, monitor colorido de 17 polegadas e sistema operacional *Windows XP*. As

experiências aconteceram em momentos distintos nos laboratórios das Faculdades Alves Faria e no laboratório de visualização da Universidade Federal de Goiás, durante os meses de março e abril de 2006.

7.1.3 Metodologia

Três grafos foram utilizados nos experimentos: o primeiro, identificado como G1, é um grafo considerado "simples" baseado no *Forrester's World Dynamics graph* [Nascimento 2003]; o segundo, identificado como G2, é um grafo mais "complexo"; o terceiro grafo, G3, foi empregado no treinamento dos usuários no sistema. Os grafos G2 e G3 foram criados exclusivamente para esta pesquisa.

Para cada grafo foi gerado um desenho inicial por meio de dez iterações do algoritmo *Spring Embedder* modificado. Esses desenhos foram salvos em arquivos e utilizados como solução inicial nos testes. A Tabela 7.1 mostra os atributos dos grafos e a qualidade de suas soluções iniciais.

Tabela 7.1: *Atributos e qualidade dos grafos do primeiro estudo piloto.*

Grafo	Vértices	Arestas	Cruzamentos	Arestas para Cima	Área
G1	43	65	24	33	554
G2	60	112	542	53	337
G3	50	69	97	34	442

O estudo piloto consistiu de três experimentos:

- Experimento (E1): doze estudantes trabalharam sozinhos no sistema *cCo-GHints*. Esses estudantes foram divididos em dois grupos de seis alunos. Cada grupo utilizou um grafo diferente (G1 ou G2). Nessa experiência não houve cooperação; cada usuário usou um computador distinto com a finalidade de melhorar um desenho de grafo inicial e não se comunicou com os demais.
- Experimento (E2): doze pares de alunos trabalharam cooperativamente no sistema *cCo-GHints*. Essas duplas também foram divididas em dois grupos de seis pares: um grupo para o grafo G1 e outro para o G2. Cada dupla trabalhou em um mesmo computador simultaneamente, podendo usar todos os recursos de colaboração da aplicação e foram, em especial, estimulados a usarem os recursos de compartilhamento e integração.
- Experimento (E3): outras doze duplas trabalharam cooperativamente no sistema *iCo-GHints*. Essas duplas foram divididos em dois grupos de seis pares para o grafo G1 e G2. Como na experiência E2, cada par trabalhou em um mesmo computador e simultaneamente.

Cada dupla ou usuário sozinho levou aproximadamente uma hora para completar a sua experiência, que foi dividida em quatro passos:

1. Um tutorial de aproximadamente 15 minutos onde foram explicadas as metas da experiência sobre desenho de grafos, os critérios estéticos a serem atingidos e os recursos da *interface* da aplicação. O tutorial mostrou aos participantes a melhor forma de usar a aplicação para se desenhar grafos.

2. Uma prática de aproximadamente 15 minutos onde os participantes utilizaram o sistema para se familiarizar com a *interface* e com o problema de desenho de grafos.

3. O experimento em si, direcionado para consumir 20 minutos. Durante essa etapa todas as ações dos usuários no sistema foram registradas automaticamente pelo sistema em arquivos de *log*. Para cada usuário foi gerado um arquivo com as informações do grafo que estava sendo utilizado, qual ação o usuário havia tomado (por exemplo, seleção de um vértice, alteração da posição de um vértice, recuperação de melhor solução, etc.) e a hora, minuto e segundo que aquela ação ocorreu. Também nessa fase os usuários foram observados sem que houvesse interferência e anotações sobre o andamento do experimento foram feitas.

4. O preenchimento de um questionário com perguntas objetivas e subjetivas sobre o experimento. Esse processo levou cerca de 5 minutos.

7.1.4 Resultados das Experiências E1 e E2

Esta seção apresenta os resultados de cada experimento do estudo piloto. Os resultados dos experimentos E1, E2 e E3 são apresentados nas Tabelas 7.2 à 7.7. Cada tabela possui um número identificador do participante ou do par de usuário, a qualidade da melhor solução obtida pelo mesmo e o tempo em minutos que essa melhor solução foi alcançada dentro do experimento, além da média e desvio padrão dos critérios de qualidade. Os experimentos realizados pelos alunos de mestrado são identificados por um asterisco (*). Comentários gerais sobre este estudo são feitos no final da seção.

A Tabela 7.2 ilustra os resultados obtidos pelos estudantes que trabalharam sozinhos na aplicação *cCo-GDHints* utilizando o grafo simples G1.

Já a Tabela 7.3 mostra os resultados das duplas que trabalharam na aplicação *cCo-GDHints* também com o grafo simples.

Comparando os valores da média do critério cruzamento de arestas (critério prioritário) encontrados nessas duas tabelas percebe-se que os usuários trabalhando

Tabela 7.2: *Resultados do Experimento E1: usuários trabalhando sozinhos na aplicação* cCo-GDHints *com o grafo simples G1.*

Usuário	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
1	13	17	359	20
2	13	34	391	20
3	16	15	437	19
4	13	18	710	16
5	17	18	553	17
6	15	18	456	15
Média	14,50	20,00	484,33	17,83
Desvio padrão	1,76	6,96	128,92	2,14

Tabela 7.3: *Resultados do Experimento E2: usuários trabalhando em dupla na aplicação* cCo-GDHints *com o grafo simples G1.*

Dupla	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
1	13	18	395	18
2	13	23	434	18
3	14	20	566	18
4	16	32	579	16
5	17	25	474	13
6	17	2	425	16
Média	15,00	20,00	478,83	16,50
Desvio padrão	1,90	10,06	76,93	1,97

individualmente na aplicação *cCo-GDHints* obtiveram melhores resultados. Pode-se verificar ainda que a melhor solução em quase todos os casos (dupla ou individual) sempre surgiu nos últimos momentos da experiência, isso se deve muito ao fator aprendizado dos integrantes da experiência, que após um determinado momento aprendiam como trabalhar melhor a solução problema.

Esses mesmos resultados se repetiram com a utilização de um grafo mais complexo. A Tabela 7.4 apresenta os resultados alcançados pelos estudantes que trabalharam sozinhos na *cCo-GDHints* com o grafo complexo G2 e a Tabela 7.5 ilustra os resultados obtidos pelas duplas no mesmo grafo complexo e aplicação.

Analisando primeiro a diferença de resultados entre a experiência E1 e E2, os dados claramente demonstram que a configuração onde usuários trabalharam sozinhos utilizando a aplicação *cCo-GDHints* mostrou ser melhor do que os usuários trabalhando em conjunto. Vários fatores observados durante os experimentos podem ter levado a esse resultado.

Um primeiro ponto observado foi a falta de coordenação. A maioria dos usuários durante o Experimento E2 não montou estratégias conjuntas para o

Tabela 7.4: *Resultados do Experimento E1: usuários trabalhando sozinhos na aplicação* cCo-GDHints *com o grafo complexo G2.*

Usuário	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
7	54	42	393	19
8	50	41	477	20
9	50	49	522	7
10	50	51	622	16
11	52	43	506	17
12	59	58	607	19
Média	52,50	47,33	521,17	16,33
Desvio padrão	3,56	6,59	85,02	4,80

Tabela 7.5: *Resultados do Experimento E2: usuários trabalhando em dupla na aplicação* cCo-GDHints *com o grafo complexo G2.*

Dupla	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
7(*)	46	53	642	19
8	53	56	1102	20
9	52	46	541	20
10(*)	61	52	1008	20
11	60	54	1238	13
12	53	64	939	17
Média	54,17	54,17	911,67	18,17
Desvio padrão	5,56	5,88	269,39	2,79

processo de otimização. Boa parte das vezes, os usuários trabalham isolados com pouca ou nenhuma interação entre os mesmos, não havendo uma efetiva coordenação das atividades de cada integrante do grupo que permitisse a eles a verificação do andamento dos trabalhos individuais e global. Segundo Dourish [Dourish 1997] a coordenação é um ponto fundamental em trabalhos colaborativos que permite ao grupo conhecer o panorama global da atividade e, assim, tomar decisões, mudar estratégias e definir os esforços da equipe. Acredita-se que a dificuldade em coordenar as atividades conjuntas no Experimento E2 foram causadas, em parte, pelo fato dos usuários estarem trabalhando em telas individuais, o que os levou a se concentrarem mais no seu trabalho individual e a interagirem menos com o outro participante.

A mudança do mapa mental também foi um dos motivos que produziu resultados diferenciados no Experimento E2. Segundo relatos dos próprios usuários, como eles podiam recuperar a melhor solução global, independente de quem a havia gerado, isso causava uma mudança na percepção que tinham do desenho, ou seja, ocorria uma alteração do mapa mental da solução corrente, quando esta havia sido criada pelo outro usuário. A recuperação de uma melhor solução global

muito diferente da solução de trabalho, levava o usuário a ter que remontar a sua estratégia para o novo desenho. Isso demandava tempo e muitas vezes frustrava o usuário, dando a impressão de ter perdido o seu trabalho anterior.

Uma outra característica observada foi a competitividade entre os usuários. Alguns usuários relataram em entrevista que, o fato do sistema *cCo-GDHints* indicar por aviso sonoro e textual qual usuário havia gerado a melhor solução de trabalho, os levou a utilizarem esse recurso para manter um processo competitivo onde cada um tentava obter resultado melhor do que o do outro participante. Neste caso, os usuários trabalhavam completamente isolados sem interação alguma com a outra pessoa.

Acredita-se também que houve dificuldade para manutenção da consciência do espaço de trabalho. Observou-se durante o Experimento E2, e depois através de entrevista com algumas pessoas, que os usuários tinham uma boa consciência a respeito do seu trabalho individual e do quanto eles poderiam melhorá-lo; contudo, não sabiam até onde poderiam contribuir para produzir uma melhor solução global. Praticamente as únicas formas de conhecimento geral sobre as atividades do grupo como um todo vinha da possibilidade de interação face-a-face ou da recuperação da melhor solução global.

Esses pontos expostos acima podem ter sido causados, em parte, pela complexidade de execução de uma atividade em grupo e, também, pelo fato da *interface* da aplicação *cCo-GDHints* não contemplar algumas funcionalidades de *groupware* descritas na Seção 4.1.5. Praticamente todos os usuários relataram ter encontrado dificuldades em interagir com a *interface* quando executavam os algoritmos de otimização, uma vez que, durante esse procedimento, o grupo precisava parar de trabalhar já que a atualização da tela ficava comprometida. Uma outra característica que, segundo os usuários, prejudicou a execução de um trabalho mais efetivo foi o tamanho da tela. Eles relataram que o pouco espaço visual oferecido pela *interface* da aplicação *cCo-GDHints* dificultava a visualização total do desenho, sendo necessário utilizar as barras de rolagem e *zoom*.

Já na experiência E1, na mesma aplicação *cCo-GDHints*, os usuários estavam sozinhos, sendo eles próprios quem coordenavam as suas atividades, montavam uma única estratégia de trabalho que era seguida até o término da experiência. A manutenção do mapa mental depois da recuperação da melhor solução global não pareceu ser uma atividade tão impactante pois, na maioria dos casos, essa solução era próxima da solução de trabalho do usuário. Contudo, convém destacar um ponto considerado primordial: observando os usuários trabalhando sozinhos percebeu-se que a concentração no problema era muito maior quando comparado aos usuários trabalhando em grupo, no Experimento E2. O fato de não existir outra pessoa na

atividade fez com que o usuário se concentrasse mais.

7.1.5 Resultados da Experiência E3

A Tabela 7.6 ilustra os resultados obtidos pelas duplas que trabalharam no grafo simples G1, mas agora utilizando a aplicação *iCo-GDHints* (Experiência E3). Comparando-a com os valores encontrados nas Tabelas 7.2 e 7.3 percebe-se que quase todos os resultados médios dos critérios estéticos diminuíram. Analisando, em especial, a média de cruzamentos de arestas, a experiência E1 obteve 14.5, a experiência E2 15.00 e a experiência E3 13.83.

Tabela 7.6: *Resultados do Experimento E3: usuários trabalhando em dupla na aplicação* iCo-GDHints *com o grafo simples G1.*

Dupla	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
13	14	32	704	10
14	14	31	541	19
15	12	28	625	19
16	17	28	682	11
17	12	42	473	14
18	14	20	371	18
Média	13,83	30,17	566,00	15,17
Desvio padrão	1,83	7,17	129,06	4,07

O mesmo ocorreu quando as duplas trabalharam no grafo complexo G2 na aplicação *iCo-GDHints*. A Tabela 7.7 ilustra os resultados e percebe-se tal fato comparando-a com os valores encontrados nas Tabelas 7.4 e 7.5. Enquanto a média de cruzamentos de arestas foi 52.50 para a Experiência E1 e 54.17 para a Experiência E2, já na Experiência E3 o resultado foi de 48.67 cruzamentos.

Tabela 7.7: *Resultados do Experimento E3: usuários trabalhando em dupla na aplicação* iCo-GDHints *com o grafo complexo G2.*

Dupla	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
19	51	50	365	16
20	50	46	559	18
21	49	52	693	14
22	47	53	667	16
23	47	54	659	17
24	48	40	600	8
Média	48,67	49,17	590,50	14,83
Desvio padrão	1,63	5,31	120,82	3,60

A Figura 7.1 mostra um gráfico que compara o número de cruzamentos (expressos em termos da média ± desvio padrão) no grafo G2 nas experiências E1, E2 e E3.

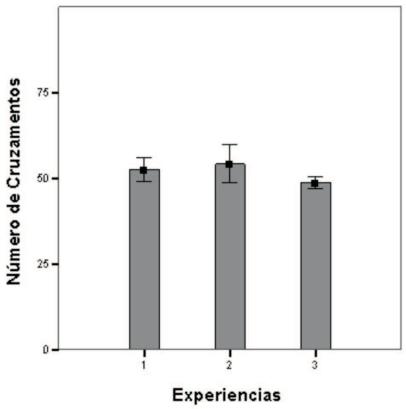


Figura 7.1: *Número de cruzamentos no grafo G2 nas experiências E1, E2 e E3. Dados expressos em média ± desvio padrão.*

Em resumo, na Experiência E3, realizada com a aplicação *iCo-GDHints*, os resultados foram melhores do que nos outros dois experimentos. Acredita-se que isso ocorreu por três motivos:

1. A montagem de estratégias e coordenação de atividades foram mais naturais. Logo no início da experiência os usuários já delimitavam áreas de atuação e combinavam o acesso aos algoritmos e recuperação das melhores soluções;

2. O contexto de trabalho era um só e os usuário pareciam ter uma consciência maior de que deveriam alcançar um objetivo comum. A percepção do que o outro estava fazendo, como poderia contribuir para a solução final e como sua ação poderia impactar no trabalho do outro era muito mais explicita;

3. Os problemas de mudança do mapa mental e competitividade foram resolvidos nesta configuração, uma vez que a solução de trabalho era única.

Contudo, um dos problemas encontrados em alguns grupos da Experiência E3 foi o conflito de interesses; apesar de ser mais fácil a coordenação de atividades, em alguns casos, os usuário possuíam intenções distintas para as ações que iriam executar. Por exemplo, um usuário queria expandir a área do desenho enquanto outro preferia trabalhar em regiões compactadas.

Na Seção 4.1.6 foram enumeradas algumas desvantagens de uma aplicação SDG citadas por Stewart *et. al.* [Stewart, Bederson e Druin 1999]. Segundo esses pesquisadores, conflitos de interesse, dificuldade de interação devido ao espaço

de trabalho reduzido e diminuição da colaboração, entre outros, podem ocorrer em atividades cooperativas que utilizam aplicações SDG. Nota-se que algumas das questões apresentadas estão em sintonia com os resultados dos experimentos, expostos anteriormente.

Convém destacar que em todas as experiências os usuários utilizaram muito pouco os operadores de integração e a área de compartilhamento. Em questionamento feito aos próprios usuários, eles relataram não terem conseguido entender a utilidades dessas ferramentas e como elas poderiam ajudá-los efetivamente a melhorar a sua solução corrente. Além de que, segundo eles, a quantidade de informação e procedimentos era muito grande para o tempo que foi dedicado ao experimentos.

7.2 Segundo Estudo Piloto

Após a análise dos resultados do primeiro estudo piloto, surgiu a curiosidade de se verificar quais seriam os resultados se o grupo utilizasse a aplicação *iCo-GDHints* em uma tela de projeção maior ao invés da tela do monitor de um computador. Montou-se então um novo experimento, identificado como E4.

7.2.1 Participantes

Neste experimento vinte e quatro estudantes do curso de graduação em Ciência da Computação da Universidade Federal de Goiás trabalharam em duplas (doze duplas), cada dupla em um mesmo computador.

7.2.2 Recursos

Os experimentos foram feitos em um computador Pentium 4, com 512Mb de memória, sistema operacional *Windows XP* e um tela de projeção vertical de aproximadamente 1,5x1,5 metros.

7.2.3 Metodologia

Essa experiência foi conduzida da mesma forma que a Experiência E3, com a diferença de que a tela era uma projeção vertical grande. Os estudantes foram divididos em doze duplas, sendo que seis trabalharam com o grafo G1 e as outras seis com o grafo G2, todos usando o sistema *iCo-GDHints*. As ações dos usuários foram registradas em arquivos de *logs* e anotações manuais foram feitas durante os experimentos.

7.2.4 Resultados da Experiência E4

A Tabela 7.8 ilustra os resultados obtidos pelas duplas que trabalharam no grafo simples G1. Comparando-a com os valores encontrados nas Tabelas 7.2, 7.3 e 7.6 verifica-se que a média de cruzamentos diminuiu: 14.5 para a experiência E1, 15.00 na experiência E2, 13.83 na experiência E3 e agora 12.33 na experiência E4. Um outro destaque foi o fato do desvio padrão para cruzamento de arestas ter sido muito baixo 0.81; isso se deve aos valores terem sido quase todos iguais.

Tabela 7.8: *Resultados do Experimento E4: usuários trabalhando em dupla na aplicação* iCo-GDHints *com o grafo simples G1 e tela grande.*

Dupla	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
25	12	26	627	10
26	14	20	524	14
27	12	33	677	18
28	12	32	648	9
29	12	25	655	19
30	12	29	587	8
Média	12,33	27,50	619,67	13,00
Desvio padrão	0,81	4,84	55,87	4,73

A redução da média de cruzamentos ocorreu também para o grafo complexo. A Tabela 7.9 ilustra os resultados obtidos pelas duplas que trabalharam no grafo G2. Através dos valores encontrados nas Tabelas 7.4, 7.5, 7.7 e 7.9 verifica-se que, para a Experiência E1, a média de cruzamentos foi 52.50, na Experiência E2 foi 54.17, na Experiência E3 alcançou 48.67 e, agora, na Experiência E4, foi 44.00.

Tabela 7.9: *Resultados do Experimento E4: usuários trabalhando em dupla na aplicação* iCo-GDHints *com o grafo complexo G2 e tela grande.*

Dupla	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
19	47	54	520	14
20	46	46	666	19
21	40	64	663	12
22	46	50	688	18
23	44	53	693	20
24	41	43	581	18
Média	44,00	51,67	635,17	16,83
Desvio padrão	2,90	7,34	69,37	3,13

Conclui-se então que o experimento E4 apresentou resultados bem melhores do que os outros três experimentos anteriores no critério cruzamento de arestas.

Esses resultados mostraram uma sintonia com os resultados das experiências executadas por Mandryk *et. al.* [Mandryk, Scott e Inkpen 2002], descritos na Seção 4.1.7, que identificaram alguns fatores que influenciam a colaboração em um ambiente de atividade em grupo, como o tamanho e o número de telas.

Segundo Mandryk *et. al.*, quanto maior o tamanho da tela maior a interação das pessoas e a percepção do problema. Acredita-se, assim, que no Experimento E4 o tamanho da tela tenha auxiliado bastante os usuários, principalmente, pela natureza do problema de otimização de desenho de grafos ser extremamente visual.

Um outro ponto que pode ter levado a resultados melhores no Experimento E4 foi o fato dele ter sido realizado com estudantes com um perfil distinto dos participantes das experiências E1, E2 e E3 (estudantes de graduação em Sistemas de Informação).

A Figura 7.2 ilustra dois estudantes em uma das experiências.

Figura 7.2: *Estudantes trabalhando em um problema de desenho de grafo usando uma tela grande.*

As Tabelas 7.10 e 7.11 resumem os dados das quatro experiências com os grafos G1 e G2 respectivamente. A Figura 7.3 ilustra o desenho inicial do grafos G2 na experiência E3 e a Figura 7.4 mostra o melhor desenho produzido pelo grupo 23[1] no mesmo experimento.

7.3 Aspetos Gerais do Trabalho

Nesta seção são apresentadas informações extras relativas à configuração dos experimentos, bem como, dificuldades enfrentadas durante o desenvolvimento das aplicações colaborativas.

[1]Desenhos contruídos por dois alunos do sexo masculino do quinto período do curso de Sistemas de Informação das Faculdades Alves Faria. A escolha desses desenho foi aleatório sem nenhuma particularidade especial.

Tabela 7.10: *Comparação dos resultados médios para o grafo simples G1.*

Experimento	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
E1	14,50	20,00	484,33	17,83
E2	15,00	20,00	478,83	16,50
E3	13,83	30,17	566,00	15,17
E4	12,33	27,50	619,67	13,00

Tabela 7.11: *Comparação dos resultados médios para o grafo complexo G2.*

Experimento	Cruzamentos	Arestas para Cima	Área	Tempo (minutos)
E1	52,50	47,33	521,17	16,33
E2	54,17	54,17	911,67	18,17
E3	48,67	49,17	590,50	14,83
E4	44,00	51,67	635,17	16,83

7.3.1 Dificuldades na Realização de Experimentos Controlados

Durante a realização dos experimentos, alguns problemas tiveram que ser contornados:

- Necessidade de um número elevado de usuários para realizar os experimentos: devido a quantidade de variáveis que compõem o problema estudado, para uma validação mais formal das hipóteses apresentadas, seria necessário um grande número de pessoas com perfis bem definidos. Contudo, encontrou-se muita dificuldade em conseguir voluntários aptos a realizar os experimentos propostos.
- Mais tempo é necessário para realizar experiências com todo o ambiente - a quantidade de informação e possibilidades de interação em cada aplicação implementada são grandes e demorariam muito mais tempo para serem completadas e analisadas nos experimentos. Por questão de prazo e pela quantidade de experiências que seriam realizadas foi necessário limitar, assim, o tempo de cada experimento em 20 minutos.
- Impossibilidade de realizar todas as experiências em um mesmo local e em horários contínuos - não foi possível realizar os experimentos em tempos contínuos. Os horários tiveram que ser agendados conforme disponibilidade dos voluntários. Isso causou uma sobrecarga de trabalho, uma vez que todas as atividades de tutorial e treinamento tiveram que ser sistematicamente repetidos. Os estudos pilotos em sua totalidade levaram cerca de dois meses para serem completados.

- Grande quantidade de dados a serem analisados: devido ao número de experimentos realizados houve a necessidade de análise de um grande volume de dados, envolvendo arquivos de *logs* e consolidação de questionários e de entrevistas com usuários.

7.3.2 Dificuldades de Desenvolvimento de Aplicação *Groupware*

A lista abaixo descreve algumas dificuldades enfrentadas durante o desenvolvimento das aplicações *cCo-GDHints* e *iCo-GDHints* e decisões de projeto tomadas.

- As duas aplicações utilizam a API *SDGToolkit* para implementar múltiplos dispositivos de *mouse*. Essa API foi desenvolvida utilizando uma arquitetura proprietária da Microsoft. Isso direcionou o desenvolvimento das aplicações *cCo-GDHints* e *iCo-GDHints* para a plataforma *Windows*, sem a possibilidade de portabilidade para outros ambientes.
- Tanto o sistema *cCo-GDHints* quanto o *iCo-GDHints* necessitam que alguns elementos de tela sejam acionados por múltiplos usuários. A API *SDGToolkit* já fornece alguns controles gráficos multiusuário, entretanto, a utilização desse recurso não é trivial. Isso exigiu um esforço maior neste trabalho para implementar componentes próprios (como botões e barras de rolagem) que se portam semelhantes aos controles normais do *Windows*.
- Um outro ponto de dificuldade foi a perda de informação visual na tela em momentos de alto processamento. Por exemplo, quando o sistema atualiza a tela para redesenhar um grafo e os usuários movem os ponteiros de *mouse*, simultaneamente, acontece um conflito entre o recurso da API, que também atualiza a tela para mostrar a nova posição dos ponteiro e os métodos de desenho do sistema. Boa parte das vezes o redesenho da tela não ocorre e toda a informação visual é perdida, até que um usuário execute alguma ação que force a atualização do desenho de grafo. Esse problema foi resolvido parcialmente através da reestruturação do código do sistema, mas as intermitências na atualização da tela ainda persiste, principalmente durante a execução dos algoritmos de otimização. Acredita-se que o problema poderia ser totalmente resolvido através de uma organização mais complexa dos objetos visuais e linhas de execução do código ou por modificação direta na API. Contudo, estas alternativas não foram testadas.

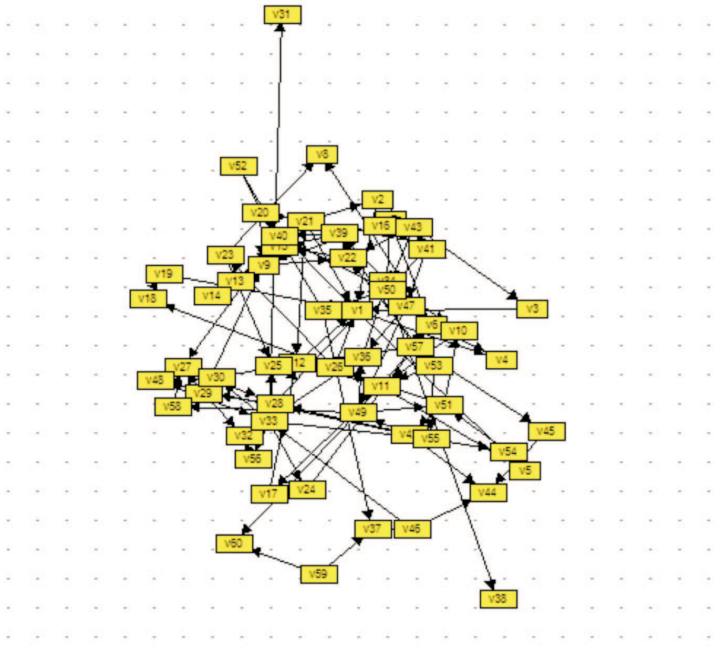


Figura 7.3: *Desenho inicial do grafo G2 do Experimento E3.*

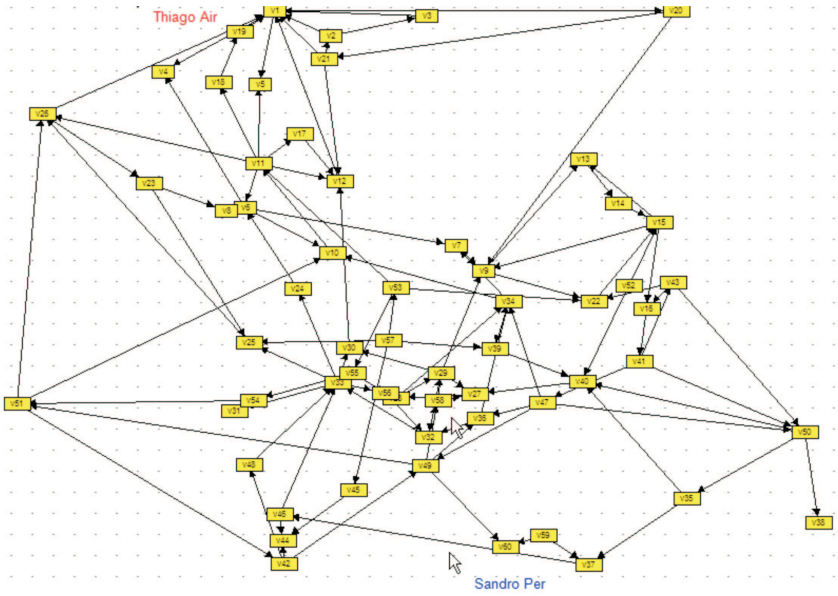


Figura 7.4: *Melhor solução do grafo G2 gerado pela dupla 23 no Experimento E3.*

CAPÍTULO 8

Conclusão

Esta dissertação apresentou uma abordagem interativa e cooperativa para problemas de otimização combinatória, chamada *Co-UserHints*, que estende o *User Hints Framework*, visando suportar a interação de mais de uma pessoa no processo de otimização. Foram definidos dentro da nova abordagem dois modelos de ambientes de trabalho cooperativo: o coletivo e o integrado. No primeiro modelo, os usuários possuem soluções individuais do problema e trabalham coletivamente para gerar uma única solução final completa. No segundo modelo, os usuários interagem desde o início com uma única solução que vai sendo melhorada cooperativamente.

Este trabalho investigou a aplicação da nova abordagem para um problema de otimização de desenho de grafos direcionados. A proposta foi estudar alternativas tecnológicas que permitissem múltiplos usuários interagir simultaneamente com métodos semi-automáticos de desenho, identificando se existe algum ganho em se inserir mais de um participante na otimização.

Dois sistemas *Single Display Groupware* de desenho de grafos foram implementados, com base nos referidos modelos e, experimentados com usuários reais em estudos pilotos.

Os resultados apontaram para um ambiente de trabalho multiusuário com uma única solução do problema e com uma área de visualização maior como a configuração mais efetiva, em comparação às propostas baseadas em usuários trabalhando sozinhos, ou em grupos, mas com soluções individuais. Experimentos mais intensivos são, contudo, necessários a fim de se comprovar essa tendência.

Durante o desenvolvimento deste trabalho alguns obstáculos foram encontrados. O primeiro deles, diz respeito ao desenvolvimento dos sistemas multiusuários. As tecnologias para a construção de aplicações *Single Display Groupware* ainda não estão solidificadas, os sistemas operacionais de uso geral não fornecem suporte nativo à utilização de múltiplos dispositivos de entrada e existem muitos requerimentos subjetivos (consciência e percepção do usuário) que devem ser levados em consideração quando da utilização de uma aplicação *groupware*. Conseqüentemente, uma grande parte do tempo da pesquisa foi dedicada a inclusão desses recursos nos

sistemas desenvolvidos.

Um segundo obstáculo foi a realização de uma grande quantidade de experimentos controlados com usuários reais. Foi difícil planejar efeivamente todos os experimentos, como também encontrar voluntários dispostos a participar dos mesmos.

8.1 Contribuições

Como contribuições deste trabalho destacam-se:

- a definição de uma abordagem multiusuário para problemas de otimização interativa geral;
- o estudo comparativo de ambientes multiusuário para desenho de grafos;
- a identificação de opções de ambientes de cooperação e variáveis que devem ser consideradas na realização de novos estudos.

8.2 Trabalhos Futuros

Alguns trabalhos futuros foram identificados para a evolução desta pesquisa são eles:

- estudar e implementar novos elementos de *interface* e de visualização de informações nos sistemas desenvolvidos – isso poderia proporcionar uma ampliação da consciência do espaço de trabalho dos usuários no problema de desenho de grafos. Tecnologias como *zoom* multiusuário, ferramentas de coordenação de atividades são algumas possibilidades;
- desenvolver e testar novos algoritmos de integração de desenhos de grafos – a proposta de operadores de integração se mostrou bastante interessante, mas o algoritmo implementado é simples e não efetivo o suficiente.
- investigar a resolução de conflitos entre restrições;
- aplicar a abordagem *Co-UserHints* para outros tipos de problemas de otimização complexos – essa abordagem investiga somente problemas de desenho de grafos. Entretanto, ela é mais genérica e pode servir para outros problemas de otimização.
- explorar a abordagem em ambientes onde os usuários estão distribuídos geograficamente – acredita-se que a *Co-UserHints* se adapta a esse tipo de ambiente sendo necessário apenas desenvolver recursos adicionais para dar suporte a colaboração remota.

Referências Bibliográficas

[Aardal et al. 1997] AARDAL, K. et al. A decade of combinatorial optimization. *CWI Tracts*, n. 122, p. 5–14, 1997.

[Aiello e Silveira 2004] AIELLO, A.; SILVEIRA, R. I. *Trazado de grafos mediante métodos dirigidos por fuerzas: revisión del estado del arte y presentación de algoritmos para grafos donde los vértices son regiones geográficas*. [S.l.], Dezembro 2004.

[Anderson et al. 2000] ANDERSON, D. et al. Human-guided simple search. *Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press / The MIT Press, p. 209–216, 2000.

[Bannon e Schmidt 1991] BANNON, L.; SCHMIDT, K. Cscw: Four characters in search of a context. *John M. Bowers, Steven D. Benford (eds.): Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, p. 3–17, 1991.

[Barbosa e Barreto 2001] BARBOSA, H. J. C.; BARRETO, A. M. S. An interactive genetic algorithm with co-evolution of weights for multiobjective problems. *Genetic and Evolutionary Computation Conference (GECCO-2001)*, Morgan Kaufmann Publishers, San Francisco, Califórnia, p. 203–210, 2001.

[Battista et al. 1999] BATTISTA, D. D. et al. *Graph Drawing: Algorithms for the Visualization of Graphs*. [S.l.]: Prentice Hall, 1999.

[Bayazit, Song e Amato 2000] BAYAZIT, O.; SONG, G.; AMATO, N. Enhancing randomized motion planners: Exploring with haptic hints. *IEEE International Conference on Robotics and Automation (ICRA'00)*, p. 529–536, Abril 2000.

[Bertault 1999] BERTAULT, F. A force-directed algorithm that preserves edge crossing properties. *7th International Symposium on Graph Drawing (GD'99)*, Springer-Verlag, London, UK, LNCS 1731, p. 351–358, 1999.

[Bertsekas 1998] BERTSEKAS, D. M. *Network Optimization: Continuous and Discrete Models*. [S.l.]: Athena Scientic, 1998.

[Blum e Roli 2003] BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268–308, 2003.

[Card, Mackinlay e Shneiderman 1999] CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. *Readings in Information Visualization: Using Vision to Think*. [S.l.]: Morgan Kaufmann Publishers, 1999.

[Davidson e Harel 1996] DAVIDSON, R.; HAREL, D. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, v. 15, n. 4, p. 301–331, Outubro 1996.

[DIAS 1998] DIAS, M. d. S. *COPSE: Um ambiente de suporte ao projeto cooperativo de software*. Dissertação (Mestrado) — COPPE/UFRJ, Rio de Janeiro, RJ, 1998.

[Dix 1997] DIX, A. Challenges for cooperative work on the web: An analytical approach. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, v. 6, n. 2-3, p. 135–156, 1997.

[Dourish 1997] DOURISH, P. Extending awareness beyond synchronous collaboration. *Workshop on Awareness in Collaboration Systems (CHI'97)*, Atlanta, Georgia, p. 22–27, Maio 1997.

[Dourish e Bellotti 1992] DOURISH, P.; BELLOTTI, V. Awareness and coordination in shared workspaces. *Conference on Computer-Supported Cooperative Work (CSCW'92)*, p. 107–114, 1992.

[Eades 1984] EADES, P. A heuristic for graph drawing. *Congressus Nutnerantiunt*, v. 42, p. 149–160, 1984.

[Ehn 1988] EHN, P. Remarks in panel discussion on cscw: What does it mean? *Conference on Computer-Supported Cooperative Work (CSCW'98)*, Portland, Oregon, p. 26–28, Setembro 1988.

[Ellis, Gibbs e Rein 1991] ELLIS, C.; GIBBS, S.; REIN, G. Groupware some issues and experiences. *Communications of the ACM*, v. 34, n. 1, p. 38–58, 1991.

[Farias et al. 2006] FARIAS, T. et al. Cida: an interaction devices management platform. *Symposium on Virtual Reality*, Porto Alegre, Belém, p. 271–284, 2006.

[Ferreira e Nascimento 2005] FERREIRA, J. de M. F.; NASCIMENTO, H. A. D. do. Otimização interativa em ambientes cooperativos. *XXXII Seminário Integrado de Software e Hardware. Anais do XXV Congresso da Sociedade Brasileira de Computação*, São Leopoldo, RS., p. 1743–1756, 2005.

[Ferreira, Nascimento e Albuquerque 2006] FERREIRA, J. de M. F.; NASCIMENTO, H. A. D. do; ALBUQUERQUE, E. S. de. Interactive optimization in cooperative environments. *Asia Pacific Symposium on Information Visualisation (APVIS2006)*, ACS, Tóquio, Japão, v. 60, p. 117–120, Fevereiro 2006.

[Fruchterman e Reingold 1991] FRUCHTERMAN, T. M.; REINGOLD, E. M. Graph drawing by force-directed placement. *Software - Practice and Experience*, v. 21, n. 11, p. 1129–1164, 1991.

[Garey e Johnson 1979] GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability - A Guide to the Theory of NPCompleteness*. [S.l.]: W.H. Freeman, 1979.

[Gershon e Page 2001] GERSHON, N.; PAGE, W. What storytelling can do for information visualization. *Communication of the ACM*, v. 44, p. 31–37, 2001.

[Goldbarg e Luna 2005] GOLDBARG, M.; LUNA, H. *Otimização Combinatória e Programação Linear*. [S.l.]: Campus, 2005.

[Greenberg, Gutwin e Cockburn 1996] GREENBERG, S.; GUTWIN, C.; COCKBURN, A. Awareness through fisheye views in relaxed-wysiwis groupware. *Graphics Interface Conference*, Morgan-Kaufmann, Toronto, Canadá, p. 28–38, Maio 1996.

[Grosz 1994] GROSZ, B. Collaborative systems. *National Conference on Artificial Intelligence (AAAI'94)*, v. 17, n. 2, p. 67–85, 1994.

[Grudin 1994] GRUDIN, J. Cscw: History and focus. *Communications of the ACM*, v. 37, n. 1, p. 92–105, 1994.

[Gutwin e Greenberg 1996] GUTWIN, C.; GREENBERG, S. Workspace awareness for groupware. *Conference on Human Factors in Computing Systems (CHI'96)*, p. 208–209, 1996.

[Gutwin, Greenberg e Roseman 1996] GUTWIN, C.; GREENBERG, S.; ROSEMAN, M. Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. *People and Computers XI (HCI'96)*, p. 281–298, 1996.

[Gutwin, Roseman e Greenberg 1996] GUTWIN, C.; ROSEMAN, M.; GREENBERG, S. A usability study of awareness widgets in a shared workspace groupware system. *Conference on Computer-Supported Cooperative Work (CSCW'96)*, p. 258–267, 1996.

[Herman, Melançon e Marshall 2000] HERMAN, I.; MELANçON, G.; MARSHALL, M. Graph visualisation and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, v. 6, n. 10, p. 24–43, 2000.

[Hertz e Widmer 2003] HERTZ, A.; WIDMER, M. Guidelines for the use of meta-heuristics in combinatorial optimization. *European Journal Of Operational Reserach*, v. 151, p. 247–252, 2003.

[Hofte 1998] HOFTE, G. H. T. *Working Apart Together - Foundations for Component Groupware*. [S.l.]: Enschede, 1998.

[Hourcade e Bederson 1999] HOURCADE, J.; BEDERSON, B. *Architecture and Implementation of a Java Package for Multiple Input Devices (MID)*. [S.l.], Maio 1999.

[Huang 1999] HUANG, M. L. *Online Information Visualization of Huge Data Spaces*. Tese (Doutorado) — Department of Computer Science and Software Engineering, The University of Newcastle, Australia, 1999.

[Inkpen et al. 2005] INKPEN, K. et al. Exploring display factors that influence co-located collaboration: Angle, size, number, and user arrangement. *HCI International*, p. 22–27, 2005.

[Inkpen et al. 1999] INKPEN, K. et al. This is fun! we're all best friends and we're all playing.: Supporting children's synchronous collaboration. *Computer Supported Collaborative Learning (CSCL'99)*, p. 252–259, 1999.

[Inkpen et al. 2001] INKPEN, K. M. et al. *Collaboration Around a Tabletop Display: Supporting Interpersonal Interactions*. [S.l.], 2001.

[Jacobsen 2001] JACOBSEN, A. E. *Interaktion und Lernverfahren beim Zeichnen von Graphen mit Hilfe evolutionarer Algorithmen (Interaction and learning methods for graph layouts with the help of evolutionary algorithms)*. Dissertação (Mestrado) — Institute AIFB, University of Karlsruhe, Karlsruhe, Alemanha,, 2001. 76128.

[Kamada e Kawai 1989] KAMADA, T.; KAWAI, S. An algorithm for drawing general undirected graphs. *Information Processing Letters*, v. 31, n. 1, p. 7–15, 1989.

[Kaufmann e Wagner 2001] KAUFMANN, M.; WAGNER, D. *Drawing Graphs Methods and Models*. [S.l.]: Springer, 2001.

[Keim 2002] KEIM, D. A. Information visualization and visual data mining. *EEE Transaction on Visualization and Computer Graphics*, v. 7, 2002.

[Khator e Leung 1997] KHATOR, S.; LEUNG, L. Power distribution planning: A review of models and issues. *IEEE Transactions on Power Systems*, v. 12, n. 5, p. 1151–1159, 1997.

[Kirsch-Pinheiro, Lima e Borges 2001] KIRSCH-PINHEIRO, M.; LIMA, J.; BORGES, M. Awareness em sistemas de groupware. *IDEAS: Centre de Información Tecnológica (CIT)*, p. 323–335, 2001.

[Kobourov e Pitta 2004] KOBOUROV, S. G.; PITTA, C. An interactive multi-user system for simultaneous graph drawing. *12th International Symposium on Graph Drawing (GD'04)*, p. 492–501, 2004.

[Leigh et al. 2002] LEIGH, J. et al. Amplified collaboration environments. *VizGrid Symposium*, Tóquio, Japão, Novembro 2002.

[Lesh et al. 2000] LESH, N. et al. *Human-Guided Search for Jobshop Scheduling*. [S.l.], 2000.

[Lesh, Marks e Patrignani 2000] LESH, N.; MARKS, J.; PATRIGNANI, M. Interactive partitioning. *8th International Symposium of Graph Drawing (GD'00)*, v. 1984, p. 31–36, Outubro 2000.

[Mame 2006] MAME. *Mame:Analog+*. 2006. Disponível em: <http://www.urebelscum.speedhost.com/, último acesso em Abril de 2006>.

[Mandryk, Scott e Inkpen 2002] MANDRYK, R.; SCOTT, S.; INKPEN, K. Display factors influencing co-located collaboration. *Conference Supplement of the ACM Conference on Computer-Supported Cooperative Work (CSCW'02)*, New Orleans, LA, USA, p. 16–20, Novembro 2002.

[Marshall 2001] MARSHALL, M. S. *Methods and Tools for the Visualization and Navigation of Graphs*. Tese (Doutorado) — University Bordeaux, 2001.

[Massie e Salisburg 1994] MASSIE, T. H.; SALISBURG, J. K. The phantom haptic interface: A device for probing virtual objects. *ASME International Mechanical Engineering Congress and Exhibition*, DSC 55-1, p. 295–302, 1994.

[Nascimento 2003] NASCIMENTO, H. A. D. do. *User Hints for Optimization Processes*. Tese (Doutorado) — University of Sydney, 2003.

[Nascimento e Eades 2001] NASCIMENTO, H. A. D. do; EADES, P. User hints for directed graph drawing. *Graph Drawing (GD2001)*, Springer-Verlag, LNCS, v. 2265, p. 205–219, 2001.

[Nascimento e Eades 2002] NASCIMENTO, H. A. D. do; EADES, P. A focus and constraint-based genetic algorithm for interactive directed graph drawing,. *Soft Computing Systems - Design, Management and Applications*, v. 87, p. 634–643, 2002.

[Nascimento e Eades 2003] NASCIMENTO, H. A. D. do; EADES, P. User hints for map labeling. *Twenty-Sixth Australasian Computer Science Conference*, Conferences in Research and Practice in Information Technology, ACS, v. 16, p. 339, 2003.

[Novák 2002] NOVáK, O. *Visualization of Large Graphs*. [S.l.], Junho 2002.

[Park 2003] PARK, K. S. *Enhancing Cooperative Work In Amplified Collaboration Environments*. Tese (Doutorado) — Graduate College of the University of Illinois at Chicago, 2003.

[Purchase, Cohen e James 1995] PURCHASE, H. C.; COHEN, R. F.; JAMES, M. Validating graph drawing aesthetics. *3th International Symposium on Graph Drawing (GD'95)*, v. 1027, p. 435–446, 1995.

[Reinhard et al. 1994] REINHARD, W. et al. Cscw tools: Concepts and architectures. *IEEE Computer*, v. 27, n. 5, p. 28–36, 1994.

[Rosete-Suarez, Sebag e Ochoa-Rodriguez 1999] ROSETE-SUAREZ, A.; SEBAG, M.; OCHOA-RODRIGUEZ, A. *A study of evolutionary graph drawing*. [S.l.], 1999.

[Scott, Carpendale e Inkpen 2004] SCOTT, S.; CARPENDALE, M.; INKPEN, K. Territoriality in collaborative tabletop workspaces. *ACM Conference on Computer-Supported Cooperative Work (CSCW'04)*, p. 6–10, 2004.

[Scott, Lesh e Klau 2002] SCOTT, S.; LESH, N.; KLAU, G. Investigating human-computer optimization. *ACM Conference on Human Factors in Computing Systems (CHI'02)*, p. 155–162, 2002.

[Scott, Mandryk e Inkpen 2003] SCOTT, S.; MANDRYK, R.; INKPEN, K. Understanding children's collaborative interactions in shared environments. *Journal of Computer Assisted Learning*, p. 220–228, 2003.

[Scott, Grant e Mandryk 2003] SCOTT, S. D.; GRANT, K.; MANDRYK, R. System guidelines for co-located, collaborative work on a tabletop display. *European Conference Computer-Supported Cooperative Work (ECSCW'03)*, Helsink, Finland, p. 14–18, Setembro 2003.

[Shen, Lesh e Vernier 2003] SHEN, C.; LESH, N.; VERNIER, F. Personal digital historian: Story sharing around the table. *ACM Interactions*, v. 10, n. 2, p. 15–22, 2003.

[Shoemaker 2000] SHOEMAKER, G. Privacy and awareness in multiplayer electronic games. *Western Computer Graphics Symposium (WCGS'00)*, Panorama Mountain Village, Canadá, p. 117–121, Março 2000.

[Shoemaker 2001] SHOEMAKER, G. B. D. *Single Display Groupware Research in the Year 2000*. [S.l.], Abril 2001.

[Sohlenkamp 1998] SOHLENKAMP, M. *Supporting group awareness in multi-user enviroment through perceptualization*. Dissertação (Mestrado) — Fachbereich Mathematik-Informatik der Universität, Fevereiro 1998.

[Stefik et al. 1987] STEFIK, M. et al. Wysiwis revised: Early experiences with multiuser interfaces. *ACM Transactions of Office Information Systems*, v. 5, n. 2, p. 147–167, 1987.

[Stewart, Bederson e Druin 1999] STEWART, J.; BEDERSON, B. B.; DRUIN, A. Single display groupware: A model for co-present collaboration. *ACM Conference on Human Factors in Computing Systems (CHI'99)*, p. 286–293, 1999.

[Sugiyama e Misue 1994] SUGIYAMA, K.; MISUE, K. *Graph Drawing by Magnetic-Spring Model*. [S.l.], Agosto 1994.

[Tamassia 1998] TAMASSIA, R. Constraints in graph drawing algorithms. *Constraints, An International Journal*, v. 3, p. 87–120, 1998.

[Tse e Greenberg 2002] TSE, E.; GREENBERG, S. Sdgtoolkit: A toolkit for rapidly prototyping single display groupware. *Extended Abstracts of CSCW 2002*, p. 173–174, 2002.

[Tse e Greenberg 2004] TSE, E.; GREENBERG, S. Rapidly prototyping single display groupware through the sdgtoolkit. *Fifth Australasian User Interface Conference*, p. 101–110, 2004.

[Tunkelang 1999] TUNKELANG, D. *A Numerical Optimization Approach to General Graph Drawing*. Tese (Doutorado) — School of Computer Science, Carnegie Mellon University, 1999.

[Wallace et al. 2004] WALLACE, G. et al. *A MultiCursor X Window Manager Supporting Control Room Collaboration*. [S.l.], Julho 2004.

[Ware 2004] WARE, C. *Information Visualization: Perception for Design*. [S.l.]: Morgan Kaufmann Publishers, 2004.

[Webber 1998] WEBBER, R. J. *Finding the Best Viewpoint for Three-Dimensional Graph Drawings*. Tese (Doutorado) — Department of Computer Science and Software Engineering, The University of Newcastle, Australia, 1998.

[West 1996] WEST, D. B. *Introduction to Graph Theory*. [S.l.]: Prentice Hall, 1996.

[Westergaard 2002] WESTERGAARD, M. Supporting multiple pointing devices in microsoft windows. *Microsoft Summer Workshop for Faculty and PhDs*, 2002.

[Zanella e Greenberg 2000] ZANELLA, A.; GREENBERG, S. *A Single Display Groupware Widget Set*. [S.l.], Março 2000.

[Zanella e Greenberg 2001] ZANELLA, A.; GREENBERG, S. Reducing interference in single display groupware through transparency. *Sixth European Conference on Computer Supported Cooperative Work (ECSCW'01)*, Setembro 2001.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo