Capítulo 3

Construção de Diagramas

Ao referir-se à exibição de diagramas de um conjunto de hiperdocumentos, identifica-se imediatamente dois problemas básicos: o número de nós pode ser arbitrariamente grande e cada nó pode ser conectado a qualquer número de outros nós. Este problema, que se refere à estrutura arbitrária de interconexões, é peculiar a grafos. Um arco é tipicamente representado por uma linha contínua que conecta dois nós. Porém, diferente dos nós, os arcos não fazem sentido sem os nós que interligam, logo não podem ser apresentados isoladamente. Por outro lado, podem cruzar nós e outros arcos, o que torna o diagrama difícil de interpretar. Um grande número de arcos a ser mostrado pode prejudicar a legibilidade, mesmo sem considerar o problema da superposição. Se múltiplos arcos forem permitidos entre um par de nós, por exemplo para significar tipos diferentes de elos, a situação piora ainda mais.

De forma geral, o problema do grande número de nós pode ser resolvido por uma janela na qual o usuário possa navegar pela rede examinando partes da estrutura em mais ou menos detalhes. Isto pode ser obtido através de um diagrama de "visão global" mostrando toda a estrutura de dados e um retângulo "você está aqui" sobreposto a visão global indicando como a subseção mostrada se relaciona com a rede inteira.

Na tentativa de solucionar o problema de um nó estar conectado a qualquer número de outros nós, foram criadas técnicas para fazer com que o diagrama se torne mais legível [Fein88] [DiMa90], usando algoritmos para layout de grafos que utilizam uma variedade de heurísticas para minimizar o número de linhas cruzando as bordas e para manter nós relacionados juntos, tentando tornar o layout mais legível e esteticamente agradável [SuTT81] [RDMM+87] [TaBB88] [GaNV88] [GKNV93].

Para obter-se diagramas úteis e legíveis, deve-se representar sua estrutura de nós e elos em vários níveis de detalhe. Por exemplo, para adquirir-se uma idéia geral de como o conjunto de informações está estruturado não se necessita de um diagrama detalhado, mas sim de um diagrama global resumido. Por outro lado, uma vez os leitores estando no meio da rede ou então já familiarizados com a estrutura global, eles necessitam de diagramas mais detalhados que mostrem todos os elos de uma dada subseção.

Para criar visões globais, os sistemas devem ter facilidades para resumir, compactar e extrair a essência do conteúdo armazenado. Porém, a geração automática de diagramas multi-níveis e a identificação dos elos importantes a serem incluídos no diagrama global não são triviais. É importante que nem todos os elos dos diagramas locais se propaguem para os diagramas globais. Para isto, mecanismos para utilizar pesos ou medidas de relevância devem ser introduzidos, baseados em similaridade de nós e interesses atuais do usuário.

A exibição seletiva de nós e arcos pode reduzir o número de objetos visíveis, independente das técnicas de distribuição espacial. Por exemplo, se nós ou arcos têm tipos associados, então, só os que satisfazem uma consulta específica sobre estes tipos deveriam ser exibidos. Se um ou mais nós ou elos são considerados pontos focais, pode-se utilizar visões com olho-de-peixe, variando seletivamente o tamanho, a posição e o estilo gráfico de nós e elos de acordo com uma função de grau de interesse, que depende da distância do nó a um ponto focal e da sua medida de importância intrínseca [Furn86].

Neste capítulo são discutidas algumas propostas para simplificação de diagramas que têm como objetivo melhorar a sua legibilidade e facilitar a orientação do usuário. Técnicas para filtragem de informação e algumas das técnicas consagradas para layout dos grafos, que serão utilizadas na construção dos browsers de documentos, serão discutidas. Para uma bibliografia mais completa sobre layout automático de grafos com o objetivo de produzir diagramas esteticamente agradáveis, deve-se consultar [BETT94].

3.1 Algoritmo de Sugiyama

Geralmente, é difícil visualizar a estrutura de um grafo quando os vértices são desenhados de uma forma não uniforme, ou quando os arcos formam caminhos não sejam facilmente seguidos pelo olho humano.

No caso de hierarquias, pode-se identificar alguns elementos que ajudam a melhorar a legibilidade do diagrama:

(a) layout hierárquico dos vértices;
(b) menor número possível de linhas (arcos) que se cruzam;
(c) linhas retas;
(d) aproximação de vértices que estão conectados e caminhos curtos;
(e) layout balanceado de arcos que chegam em um vértice ou saem dele, significando que os caminhos de entrada e saída de um vértice sejam desenhados claramente.

A partir desses elementos, pode-se estabelecer algumas regras para desenhar a estrutura hierárquica:

- vértices devem ser posicionados em linhas horizontais para cada nível da hierarquia sem haver sobreposição;

- cada arco deve ser desenhado como uma linha reta.

Essas duas regras satisfazem os elementos (a) e (c) respectivamente. Para satisfazer os outros, o algoritmo de Sugiyama sugere quatro passos:

1. deve ser definida um hierarquia adequada, posicionando todos os nós em níveis. Se houver arcos muito longos, ou seja, que cruzam mais de um nível da hierarquia, são inseridos alguns vértices e arcos virtuais. Desta forma, os arcos só mudam de direção na posição desses vértices, dando origem a outros segmentos de arco. Os nós virtuais não

têm tamanho ou nome, porém são tratados como nós comuns pelo algoritmo (vide Figura 23);

2. o número de arcos que se cruzam pode ser reduzido permutando a ordem de vértices em cada nível (vide Figura 24);

3. as posições horizontais dos vértices são determinadas considerando os elementos (c), (d) e (e). A ordem dos vértices obtida no passo 2 deve ser respeitada, dando ao diagrama um layout mais simples de ser entendido;

4. um diagrama bidimensional da hierarquia é automaticamente desenhado onde os vértices e arcos virtuais criados no passo 1 são eliminados e os arcos longos correspondentes são regerados (vide Figura 25).

A fase de permutação dos nós em cada nível para diminuir o número de arcos que se cruzam é feita percorrendo-se o grafo várias vezes. Da primeira vez, percorre-se o grafo a partir do topo e, da segunda, a partir do último nível. Os passos subseqüentes se alternam entre o percorrimento top-down e bottom-up até o algoritmo terminar.

O número que estima a melhor posição horizontal do nó é chamado de baricentro, e é usado para ordenar os nós em um nível. O algoritmo usa dois baricentros para cada nó: o baricentro superior e o inferior. O baricentro superior (inferior) é a posição média entre os predecessores (sucessores) do nó incluindo nós virtuais no nível anterior (posterior). Permutando os nós em determinado nível pelos seus baricentros superiores (inferiores), minimiza-se o número de cruzamento de arcos com o nível anterior (superior).

No passo 3 do algoritmo, são determinadas as posições x e y finais de cada nó. Primeiramente, os nós são distribuídos em níveis e estes são separados por uma distância uniforme. Depois, percorre-se o grafo várias vezes para alinhar os arcos longos, eliminando os vértices virtuais. A posição dos nós pode ser modificada porém sem alterar sua ordem em cada nível.

Para maiores detalhes sobre os algoritmos utilizados para obter os elementos desejáveis em grafos legíveis, deve-se consultar [SuTT81].

No caso do sistema GRAB (Seção 2.9), este algoritmo foi modificado para lidar com ciclos e melhorar a disposição final do diagrama, incluindo a possibilidade do usuário definir constantes no layout, por exemplo, posicionando um nó no topo do grafo.

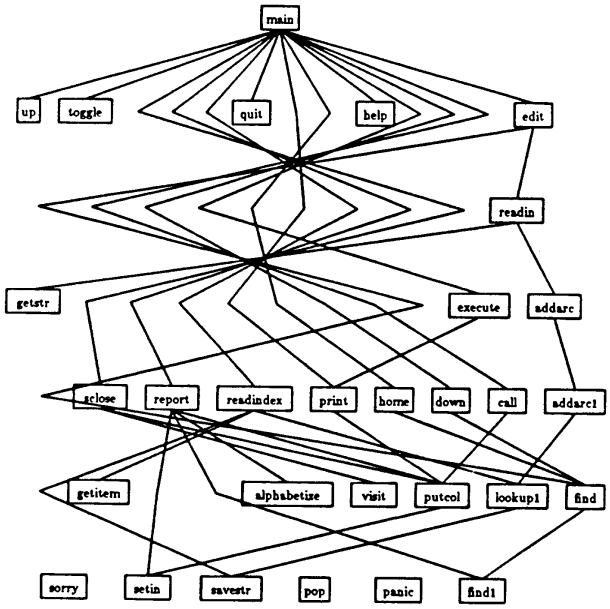

Figura 23: GRAB - Diagrama do grafo após o passo 1 do algoritmo

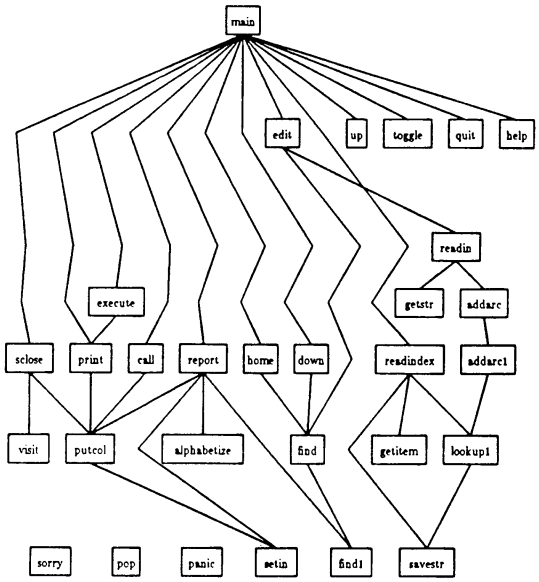

Figura 24: GRAB - Diagrama do grafo após o passo 2 do algoritmo

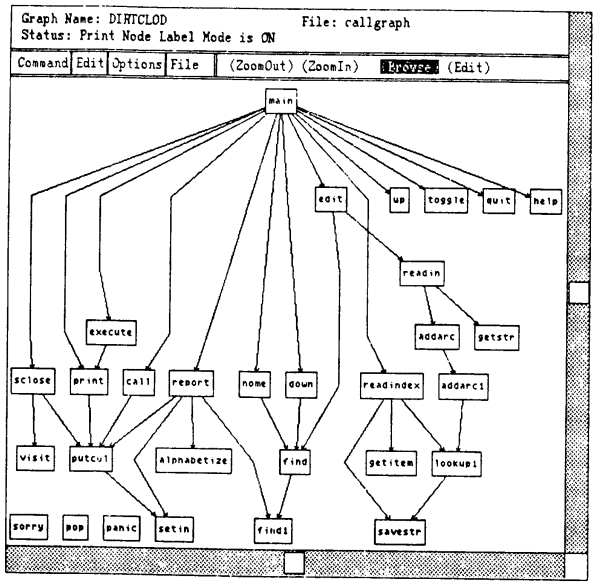

Figura 25: GRAB - Diagrama do grafo após os passos 3 e 4 do algoritmo

Dentre as modificações feitas pelo sistema GRAB [RDMM+87] no algoritmo de Sugiyama, pode-se ressaltar:

- no caso de haver ciclos no grafo, o que não é tratado pelo algoritmo original, esses serão "quebrados" da seguinte forma: os arcos que completam o ciclo têm seu sentido temporariamente invertido. Depois do diagrama desenhado, os arcos invertidos mudam novamente de sentido, voltando a direção correta;

- na fase de permutação dos nós em um nível para minimizar o número de cruzamento entre arcos, depois de percorrer o grafo a partir do topo (calculando o baricentro superior) e a partir do último nível (calculando o baricentro inferior), os próximos percorrimentos utilizam a média desses dois baricentros. Isto melhora o caso em que a posição do nó determinada por um baricentro é muito diferente da determinada pelo outro;

O maior problema do algoritmo é o tempo gasto durante a computação, que fica insuportável para grafos muito grandes. Por isso, foi desenvolvido um outro algoritmo que decompõe o grafo em uma coleção de subgrafos. Estes subgrafos são desenhados separadamente, mas os arcos que conectam nós em subgrafos diferentes também são exibidos. A vantagem desta decomposição é que ela reduz o problema do tamanho sem prejudicar a qualidade do diagrama. Esta foi a técnica utilizada pelo sistema Compoze [MeRH91], mencionado na Seção 2.9.

3.2 Notação por Conjuntos Aninhados

A técnica tradicional para representar estruturas em árvore é através de um grafo dirigido, com o nó raiz no topo da página e os nós filhos abaixo dos pais com linhas os conectando (vide Figura 26). Knuth [Knut73] tem uma longa discussão sobre esta representação padrão,

especialmente questionando por que a raiz está no topo e oferece várias alternativas de representação, incluindo a notação por conjuntos aninhados.

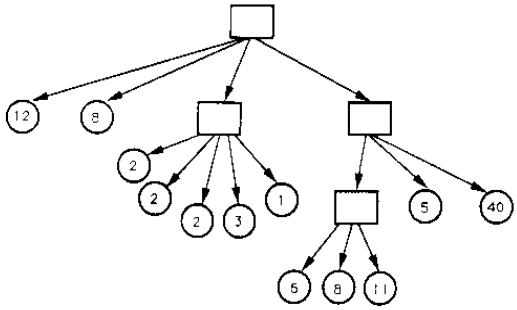

Figura 26: Notação tradicional para representar árvores

A notação por conjuntos aninhados expressa a hierarquia de uma árvore através de conteúdo espacial. Sendo assim, cada nó é colocado dentro do seu nó pai, e o conteúdo dos nós representa a hierarquia desejada. Este tipo de representação é bastante interessante para estruturas que definem tanto relações hierárquicas como relações de referência cruzada, pois os tipos de relacionamento são representados de forma diferenciada: a hierarquia através de conteúdo espacial e os elos através de ligações entre os nós (vide Figura 27).

O IGD (Seção 2.10) adotou uma variante da notação por conjuntos aninhados utilizando três técnicas básicas para esconder algumas informações e desenhar o layout final:

- supressão de detalhes de sub-árvore: não apresenta a estrutura interna dos nós filhos do nó selecionado;

- seleção de sub-árvore no mapa: o usuário pode navegar em profundidade pela hierarquia, selecionando um nó visível em qualquer nível dentro do nó mais externo apresentado em uma janela;

- herança de elos: os elos são mostrados diretamente só quando conectam nós dentro do mesmo nó-pai. Ao apresentar um elo entre dois nós, é procurado o primeiro ancestral comum dos nós, que, no limite, é o próprio documento. O arco é desenhado ligando os dois filhos do ancestral comum que estão no caminho para os dois nós originais. Já que os elos são desenhados somente entre filhos do mesmo pai, arcos visíveis nunca cruzam as fronteiras do nó-pai, fazendo com que cada janela contenha um nó inteiro, eliminando o problema de origens e destinos fora dos limites da janela (vide Figura 27).

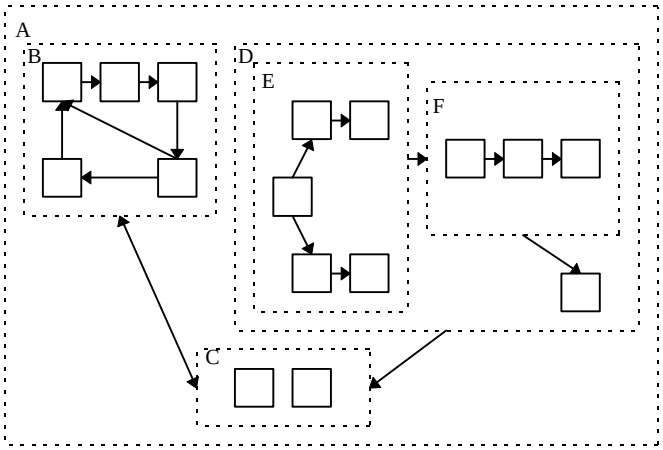

Figura 27: IGD - notação por conjuntos aninhados com herança de elos

3.3 Desenhando Grafos Compostos

Na maioria dos estudos sobre layout automático de grafos, os tipos de grafos tratados são sempre os grafos dirigidos e não-dirigidos comuns, onde somente se considera relações de adjacência entre os nós.

Em grande parte das aplicações, incluindo o propósito principal deste trabalho, é necessário o tratamento de estruturas de diagramas que representem não só adjacência entre nós, mas também relações de inclusão.

Nesta seção, será apresentado um método para desenhar grafos dirigidos compostos com a possibilidade de representação de relações de inclusão e adjacência entre nós [SuMi91]. Este método foi implementado no sistema D-ABDUCTOR [Misu94] que oferece uma série de facilidades para o tratamento de tais grafos, sendo o primeiro sistema a oferecer esses recursos baseando-se no layout automático de grafos.

Na técnica de layout, as composições são representadas por relações de inclusão entre retângulos e os arcos de adjacência por setas conectando os pares de vértices correspondentes.

Os elos de inclusão são representados por conteúdo espacial no diagrama e o cruzamento de elos é reduzido ao mínimo possível.

O algoritmo para desenhar um grafo composto consiste de quatro passos:

1. Hierarquização: quando é dado um grafo, checa-se a possibilidade de hierarquização. Se não for possível, encontramos arcos de adjacência inversos para associar a cada vértice um nível de composição;

2. Normalização: o grafo obtido no passo 1 é convertido trocando todos os arcos de adjacência impróprios por grafos dirigidos apropriados. A complexidade de tempo deste passo é proporcional ao número total de arcos de adjacência e inclusão;

3. Ordenação dos vértices: em cada hierarquia local a ordem horizontal dos vértices é determinada permutando-se a ordem dos vértices, tendo como objetivo melhorar a proximidade de vértices e o cruzamento de linhas;

4. Layout métrico: a posição horizontal dos vértices é definida considerando os critérios de proximidade, linhas retas e regras de balanceamento. A ordem dos vértices obtida no passo 3 é considerada para preservar o número reduzido de cruzamento de arcos. Um mapa é automaticamente desenhado, quando a orientação original dos arcos invertidos é restaurada e os vértices e arcos virtuais são eliminados.

Através de testes verificou-se que o algoritmo apresenta desempenho bastante satisfatório, sendo uma importante ferramenta para tratamento de grafos com composições. Verificou-se também que a legibilidade dos diagramas pode ser melhorada utilizando-se linhas curvas [SuMi91] (vide Figura 28).

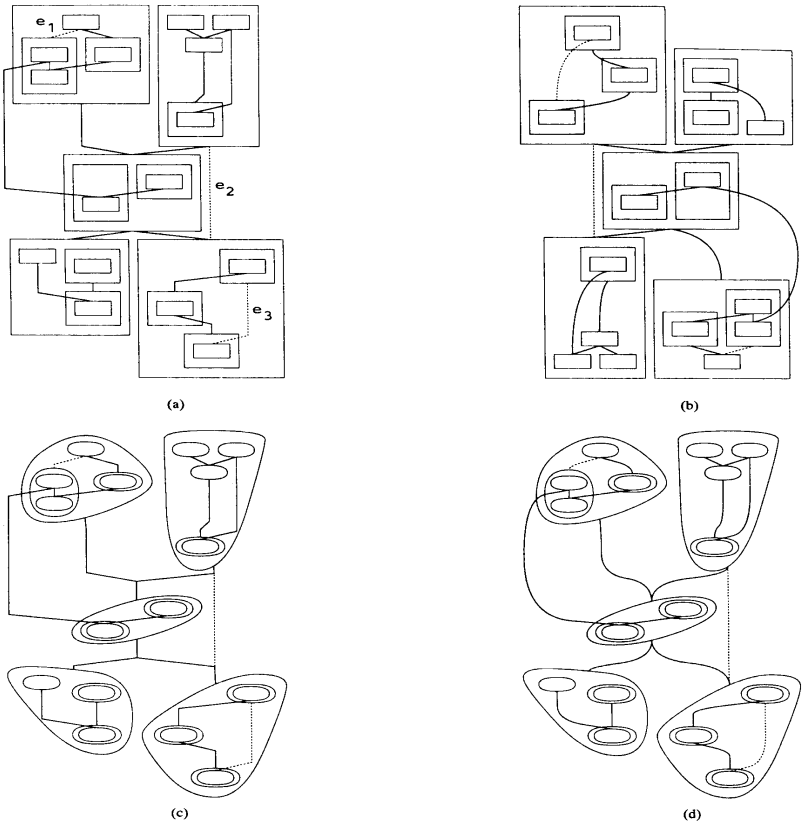

Figura 28: Variações no desenho de um mapa composto

3.4 Conceito de HYPERDRAWERS

Alguns autores afirmam que diagramas não podem ser produzidos automaticamente devido a duas razões principais:

- uma boa estrutura de grafo para uma projeção bidimensional de uma rede de hipertexto multidimensional é difícil de encontrar. Em particular, esta tarefa é praticamente impossível sem saber o conteúdo do documento;

- é impossível desenhar mais de vinte ícones de nós em uma tela. Para todo documento que contiver mais nós, muitos deles devem ser suprimidos para que a tela não se torne ilegível. Isto requer conhecimento sobre o conteúdo do documento e as intenções e objetivos do leitor.

Baseando-se nestas duas afirmações, surge o conceito da geração automática de elos dinâmicos, que não utiliza a estrutura de nós e elos do hipertexto, mas analisa o conteúdo dos nós e computa uma nova estrutura de grafo para o diagrama, definindo os chamados HYPERDRAWERS. Estes últimos formam os CYBERMAPS que podem ser usados para complementar os mecanismos de navegação dos sistemas (vide Seção 2.11) [Gloo91].

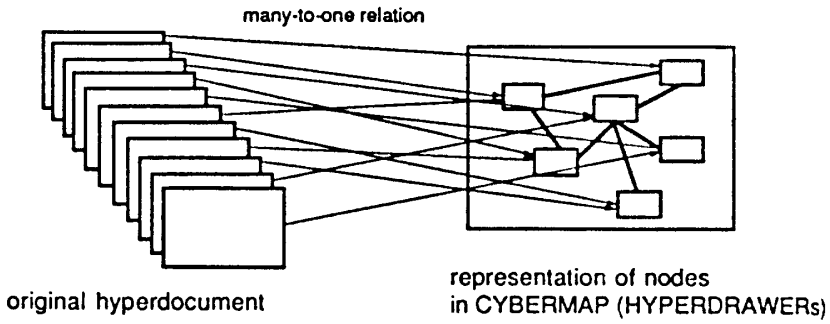

Figura 29: CYBERMAP - Relação entre nós de um documento e HYPERDRAWERS

O CYBERMAP ideal está sempre presente toda vez que o leitor acessa o documento. Ele serve como um ponto de partida fornecendo ao leitor uma idéia gráfica do hipertexto e sua posição corrente, refletindo todos os movimentos de navegação.

Um retângulo na janela do CYBERMAP representa não só um nó, mas uma coleção de nós relacionados, chamada de HYPERDRAWER (vide Figura 29). Este conceito fornece meios de transformar o hipertexto em um conjunto de fragmentos menores e manuseáveis. Um nó pertence a apenas um HYPERDRAWER. O HYPERDRAWER é equivalente à noção de cluster na recuperação de informações.

O algoritmo para a geração automática do CYBERMAP é baseado na estratégia de olho-de-peixe [Furn86], que será abordada na próxima seção.

O CYBERMAP oferece não só uma mapa global do hipertexto, mas uma visão personalizada baseada no modelo do usuário. Isso significa que o sistema deve manter duas bases de conhecimento sobre o usuário:

- o perfil do usuário, contendo dados sobre seus interesses e preferências. Esses dados são manipulados diretamente pelo usuário e podem ser modificados durante toda a sessão;

- a história da leitura do usuário, que mostra pelo menos onde ele já esteve. Uma versão avançada do sistema deveria ser capaz de atualizar o modelo do usuário considerando o seu comportamento de leitura.

Para particionar os nós em HYPERDRAWERs, existem três problemas que devem ser solucionados:

- colocar nós relacionados no mesmo HYPERDRAWER. Os HYPERDRAWERs devem ser identificados e nós relacionados devem ser posicionados juntos. Para isto, é necessário identificar uma função de hash F com as seguintes propriedades ideais:

    - F deve distribuir os nós igualmente nos HYPERDRAWERs;
    - F deve garantir que não existam mais de vinte HYPERDRAWERs, senão a tela fica muito carregada;
    - F deve agrupar os nós mais relacionados, o que significa que o conteúdo de cada nó deve ser analisado e que uma medida de afinidade entre eles deve ser definida;

- dar nomes significativos aos HYPERDRAWERs;

- identificar a topologia. A estrutura do documento deve ser baseada na análise do conteúdo dos nós, não tendo relação alguma com os elos existentes no hipertexto.

A análise de conteúdo do documento é baseada em um índice, que pode estar disponível ou ser gerado automaticamente. Ele é utilizado para geração automática da estrutura de grafo do hipertexto, baseada na ocorrência de termos do índice nos nós do documento, e para geração de uma lista de interesse, onde o usuário pode selecionar os termos mais relacionados aos seus objetivos.

Para melhorar o layout, os HYPERDRAWERs mais relacionados são posicionados no centro da tela, enquanto os menos relacionados são colocados longe do centro.

O perfil do usuário é utilizado para reduzir o número de nós exibidos no CYBERMAP, só os que são de interesse do leitor são mostrados. Depois de cada modificação do perfil do usuário, a representação do mapa é novamente computada.

Os documentos ligados dinamicamente não possuem nenhuma âncora pré-definida, ao invés disso, os elos são computados em tempo de execução de acordo com as necessidades do usuário. Isto oferece duas vantagens:

- uma maneira mais flexível de navegar no hipertexto, pois os elos são gerados de acordo com cada leitor;

- facilidade para alterações, pois criar ou eliminar nós não implica em mudanças na estrutura de elos que afetem outros nós do documento.

A diferença principal entre o CYBERMAP e os mecanismos de construção de diagramas dos sistemas hipermídia é que estes se baseiam em elos estáticos enquanto o CYBERMAP se baseia em elos dinâmicos.

Este modelo só não é adequado para hipertextos extremamente grandes com milhares ou até milhões de nós, pois existe apenas um CYBERMAP para o documento inteiro. Isto poderia ser solucionado com a existência de sub-CYBERMAPs, resultando em uma hierarquia de CYBERMAPs.

3.5 Visões com Olho-de-Peixe

Em muitos contextos, os seres humanos geralmente representam o que está em seu redor com muito detalhe, mas o que está mais distante somente através de pontos de referência. Isso sugere que tais representações que utilizam as chamadas visões de olho-de-peixe, sejam usadas para exibição de grandes estruturas de informação, tais como hipertextos. Entende-se por visão a maneira como se observa um objeto, por exemplo, um nó no diagrama. Visões com olho-de-peixe são representações que utilizam a técnica de lentes de olho-de-peixe para serem construídas, baseando-se no nó que está sendo observado. Nesta técnica, as regiões próximas ao nó observado e as marcadas como de grande importância são mostradas em detalhes, ao passo que as regiões distantes são mostradas de forma compactada, resumida. A noção de proximidade, de suma importância na técnica, deve ser rigorosamente definida.

Geralmente, os sistemas que não usam o conceito da visão olho-de-peixe apresentam detalhes locais e dados de âmbito global ao mesmo tempo, o que aumenta a complexidade do diagrama, pois a quantidade de informação é muito grande. De maneira geral, isto é implementado através de uma simples janela que permite a exibição da estrutura centrada em algum ponto. O usuário navega pela estrutura movendo a janela através de scroll ou percorrendo arcos. Como resultado, ele acaba ficando perdido, ou seja, encontrando-se em algum lugar errado e sem saber como chegar ao local certo.

Idealmente, representações com olho-de-peixe fornecem detalhes locais e globais sobre onde o usuário se localiza na estrutura total. Porém não exibe todas as informações existentes. Isto é bastante útil nos browsers, pois permite saber a posição no contexto global e quais conceitos e informações estão relacionadas à informação sendo acessada no momento.

Como exemplo da visão olho-de-peixe, pode-se observar as Figuras 30 e 31 que exibem o mesmo programa em linguagem “C” visto de duas formas. A primeira mostra a visão que é geralmente fornecida nos editores de texto, exibindo muitos detalhes que talvez não sejam importantes quando o usuário está trabalhando na linha indicada por “>“. A segunda mostra tanto detalhes locais à linha que está sendo trabalhada, como informações de contexto global, tornando-se mais interessante e até mais útil. Por esta visão, observa-se que o comando “case” está inserido em um comando “if”, que por sua vez está dentro de um “while”, que está no programa “main”.

```
 28                          t[0] = (t[0] + 10000)
 29                               - x[0];
 30                          for(i=1;i<k;i++){
 31                               t[i] = (t[i] + 10000)
 32                                    - x[i]
 33                                    - (1 - t[i-1]/10000);
 34                               t[i-1] %= 10000;
 35                          }
 36                          t[k-1] %= 10000;
 37                          break;
 38                     case 'e':
>39                          for(i=0;i<k;i++) t[i] = x[i];
 40                          break;
 41                     case 'q':
 42                          exit(0);
 43                     default:
 44                          noprint = 1;
 45                          break;
 46                }
 47                if(!noprint){
 48                     for(i=k - 1;t[i] <= 0 && i > 0;i--);
 49                     printf("%d",t[i]);
 50                     if(i > 0) {
```

Figura 30: Visão padrão de um Programa “C”

```
  1 #define DIG 40
  2 #include <stdio.h>
...4 main()
  5 {
  6      int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
...8      while((c=getchar()) != EOF){
  9           if(c >= '0' && c <= '9'){
..16           } else {
 17                switch(c){
 18                     case '+':
..27                     case '-':
..38                     case 'e':
>>39                          for(i=0;i<k;i++) t[i] = x[i];
 40                          break;
 41                     case 'q':
..43                     default:
..46                }
 47                if(!noprint){
..57                }
 58           }
 59           noprint = 0;
 60      }
 61 }
```

Figura 31: Visão com Olho-de-Peixe de um Programa “C”

A estratégia básica do olho-de-peixe utiliza uma função de grau de interesse (DOI - Degree of Interest), que atribui a cada nó do hipertexto um número representando o grau de interesse do usuário, dado o contexto atual.

Esta função pode ser decomposta em duas componentes:

- a primeira é chamada de importância à priori (API - a priori importance) e descreve a importância de um nó x para o nó atual y de interesse do usuário;

- a segunda componente determina a distância (D) entre o nó x e o nó atual y.

A essência da visão olho-de-peixe é que a função DOI aumenta com a importância à priori e diminui com a distância:

$$DOI(x,y) = API(x,y) - D(x,y),$$

onde DOI é o grau de interesse do usuário em um ponto x, dado que o ponto de foco corrente é y.

No caso de estruturas hierárquicas, tem-se $API(x,y) = - D(x,raiz)$, o negativo da distância de *x* à raiz. O sinal de menos indica uma relação inversa entre a distância da raiz e a importância do nó, quanto mais longe da raiz, menor a sua importância.

$$DOI(x,y) = - (D(x,y) + D(x,raiz))$$

Escolhendo um determinado valor *k* e só exibindo os pontos *x* que possuem $DOI(x,y) >= k$, pode-se obter visões olho-de-peixe em vários níveis de detalhe.

Esta estratégia, introduzida por Furnas [Furn86], é aplicável a estruturas onde os componentes necessários à função olho-de-peixe podem ser definidos, como por exemplo, listas, árvores, estruturas hierárquicas e outras mais.

Essas visões têm várias propriedades interessantes. Em árvores, pode-se ressaltar:

- a visão atinge uma ordem logarítmica;

- existem algoritmos rápidos para computá-las com tempo proporcional ao tamanho da visão, e não ao da árvore original;

- mudando o ponto focal, a visão é facilmente recalculada, já que a função DOI acima do ancestral comum não é modificada;

- os usuários podem navegar pela estrutura em um número de passos proporcional ao logaritmo do número de folhas consideradas, dependendo do valor k.

Infelizmente, existe uma incompatibilidade desse conceito com hipertextos não estruturados, que pode ser revelada por duas razões principais:

- a possível existência de mais de um caminho do nó atual a um outro nó do hipertexto. Além disso, quando ciclos são permitidos, a questão de como determinar um valor único para a distância não pode ser respondida; e

- a falta da raiz em documentos não estruturados, o que leva ao problema de escolher um ponto de referência necessário para determinar a importância à priori.

Pode-se tentar adaptar essa visão para hipertextos não estruturados [ToDi92], através de duas estratégias. O princípio básico da primeira estratégia consiste em utilizar o máximo possível da informação dada para penetrar na estrutura do documento ou na relação entre os nós, por exemplo, através do nome ou conteúdo do nó. A segunda estratégia, ao invés de deduzir a informação semântica pela sintaxe, solicita que os autores modelem a informação semântica diretamente no hipertexto, por exemplo, fornecendo pesos aos elos. Apesar disto aumentar a sobrecarga cognitiva [Conk87], vários benefícios são obtidos. Quanto maior é o peso de um elo, mais próxima é a relação semântica entre os nós conectados. Com este recurso, os autores podem tornar o seu conhecimento sobre a relação entre os nós acessível aos usuários subseqüentes.

3.5.1 Visões Gráficas de Grafos com Olho-de-Peixe

Como foi visto na seção anterior, uma visão de um grafo com olho-de-peixe mostra uma área grande de interesse com bastante detalhe e o resto sucessivamente menor e com menos detalhes.

Em [SaBr92], são introduzidas considerações de layout no formalismo do olho-de-peixe, tal que a posição, tamanho e nível de detalhe dos objetos exibidos são computados baseados na função de grau de interesse. Na formulação original de Furnas, ou um componente está presente com todos os detalhes ou está completamente fora da visão, não existindo controle explícito sobre o layout gráfico.

Ao gerar um visão com olho-de-peixe segundo [SaBr92], é necessário aumentar os vértices (nós) de maior interesse e diminuir os nós de menor interesse. Além disso, as posições de todos os nós também devem ser recomputadas para alocar mais espaço para a parte que foi aumentada com o objetivo de que toda a visão continue a ocupar o mesmo espaço da tela (vide Figura 33).

Intuitivamente, a posição de um nó na visão olho-de-peixe depende da sua posição na visão normal e da sua distância ao nó em foco, onde a distância é dada pela distância euclideana entre os vértices na visão normal. O tamanho do nó na visão olho-de-peixe depende da sua distância ao foco, do seu tamanho na visão normal e da sua importância intrínseca, que é dada pelo nível do nó na hierarquia (por exemplo, em um layout hierárquico, a raiz tem a maior API e as últimas folhas da árvore têm a menor). A quantidade de detalhe exibido em um nó depende do seu tamanho na visão olho-de-peixe.

O processo de geração a visão olho-de-peixe é constituído de dois passos. Primeiramente, aplica-se uma transformação geométrica à visão normal para reposicionar os vértices e aproximar ou afastar áreas próximas ou distantes do foco, respectivamente. O segundo passo é utilizar a API dos vértices para obter seu tamanho final e então desenhar o layout da visão.

Note que na visão olho-de-peixe de Furnas, cada nó pode ser exibido ou omitido, não existindo uma maneira explícita para variar o tamanho dos nós e o nível de detalhe visual.

Nesta proposta, a função de grau de interesse foi trocada pela função de valor visual do nó, que oferece considerações de layout na visão olho-de-peixe.

Os autores Sarkar e Brown [SaBr92] sugerem algumas funções que foram utilizadas na implementação de um protótipo, que exibe a visão com olho-de-peixe de um grafo e a atualiza em tempo real (vide Figuras 32, 33 e 34). Esta técnica modifica um layout existente através da estratégia de olho-de-peixe aplicada à posição e ao tamanho dos vértices.

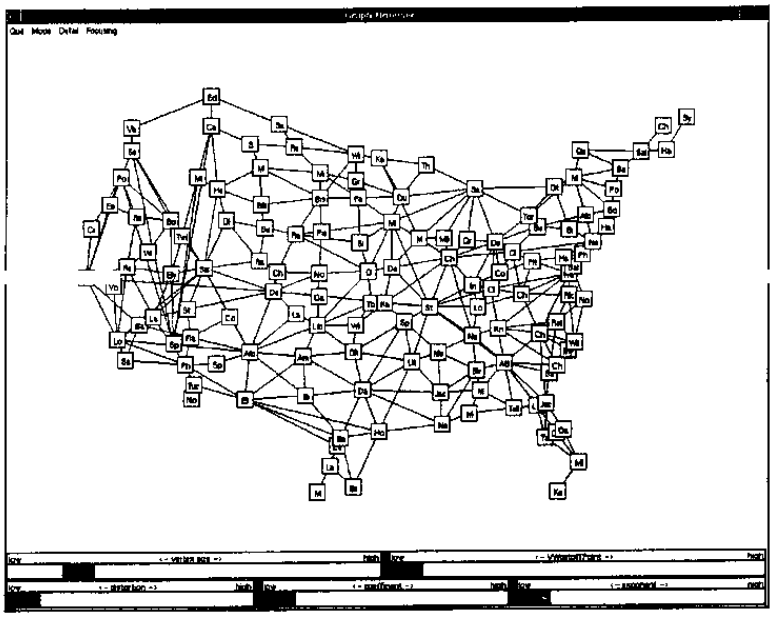

Figura 32: Layout inicial de um grafo com 134 vértices e 338 elos

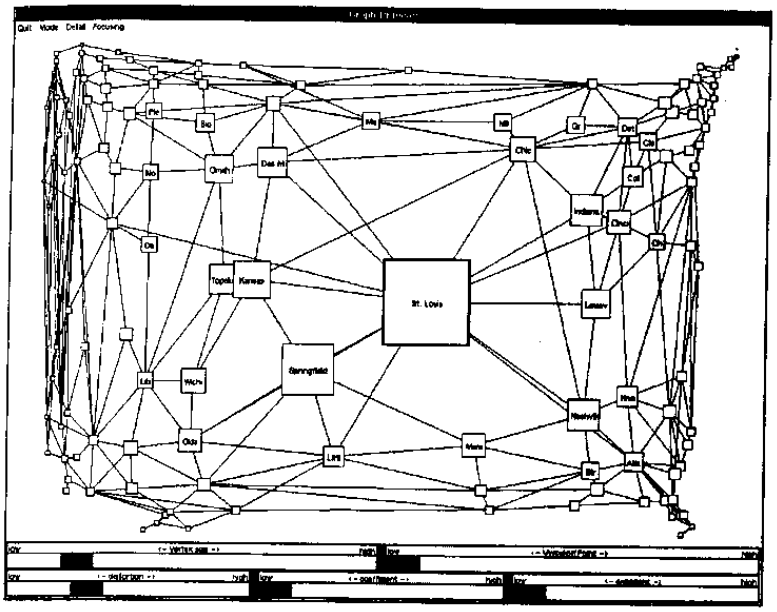

Figura 33: Uma visão com olho-de-peixe da Figura 32

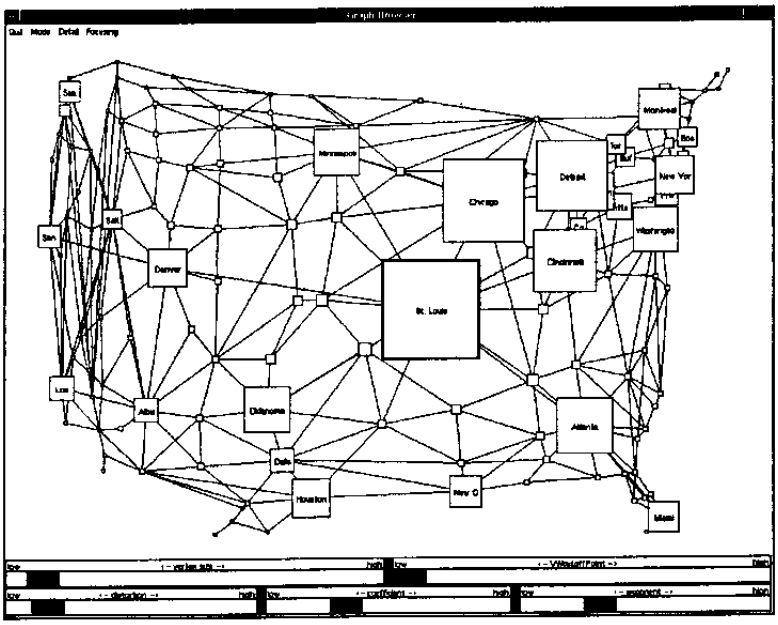

Figura 34: Outra visão com olho-de-peixe da Figura 32
mudando os parâmetros das funções escolhidas

3.5.2 Visões com Olho-de-Peixe de Redes Aninhadas com Múltiplos Pontos Focais

Sarkar e Brown [SaBr92] descreveram uma visão com olho-de-peixe de grafos, onde as posições e tamanhos dos nós eram alteradas. Porém, esta solução estava limitada a grafos não aninhados com um ponto focal único. Esta seção descreve uma técnica para gerar visões com olho-de-peixe de grafos aninhados hierarquicamente com pontos focais múltiplos e variáveis [Noik93].

Em [SaBr92], a distância entre nós é computada pela proximidade dos mesmos em um layout inicial, esta consideração só é efetiva quando a distância abstrata entre os nós corresponde à sua proximidade no desenho (por exemplo, no caso de redes de transportes onde os nós são posicionados de acordo com dados cartográficos). A técnica a ser descrita a seguir ultrapassa a limitação da distância abstrata entre nós corresponder à sua proximidade no desenho.

Sabe-se que uma visão global com apenas um nível de visualização não é muito eficaz, tanto para navegação como para exibição da estrutura do documento. Utilizando técnicas de agregação de nós e herança de elos, consegue-se melhorar a visão geral da estrutura, porém se o tamanho do espaço de informações aumentar muito, são necessárias técnicas adicionais para a exibição do mapa.

Na técnica em questão, o crucial é gerar visões com olho-de-peixe modificando o tamanho de cada elemento em relação a seu DOI antes de posicioná-lo por um algoritmo de layout ao invés de alterar um layout existente.

Por propósitos de layout, a geometria de um vértice *v* é especificada em relação a geometria de seu pai. Isto torna possível modificar o tamanho dos vértices na sub-árvore de *v* simplesmente modificando o tamanho de *v*. Ao desenhar os grafos aninhados, a estrutura de aninhamento é exibida através de conteúdo espacial. Para gerar uma visão com olho-de-peixe, adapta-se o tamanho de *v* de acordo com a função DOI, concordando com a interpretação de que o tamanho de um elemento na tela reflete seu grau de interesse pelo usuário (vide Figura 35).

A proposta apresentada em [Noik93] ignora os elos da estrutura para definir a proximidade dos nós e calcular a função de grau de interesse, e não considera a distância dos nós ao nó corrente (localização do usuário no hiperdocumento). A função de grau de interesse é computada em relação à estrutura hierárquica e verificando se as composições contêm os pontos focais ou estão contidas neles.

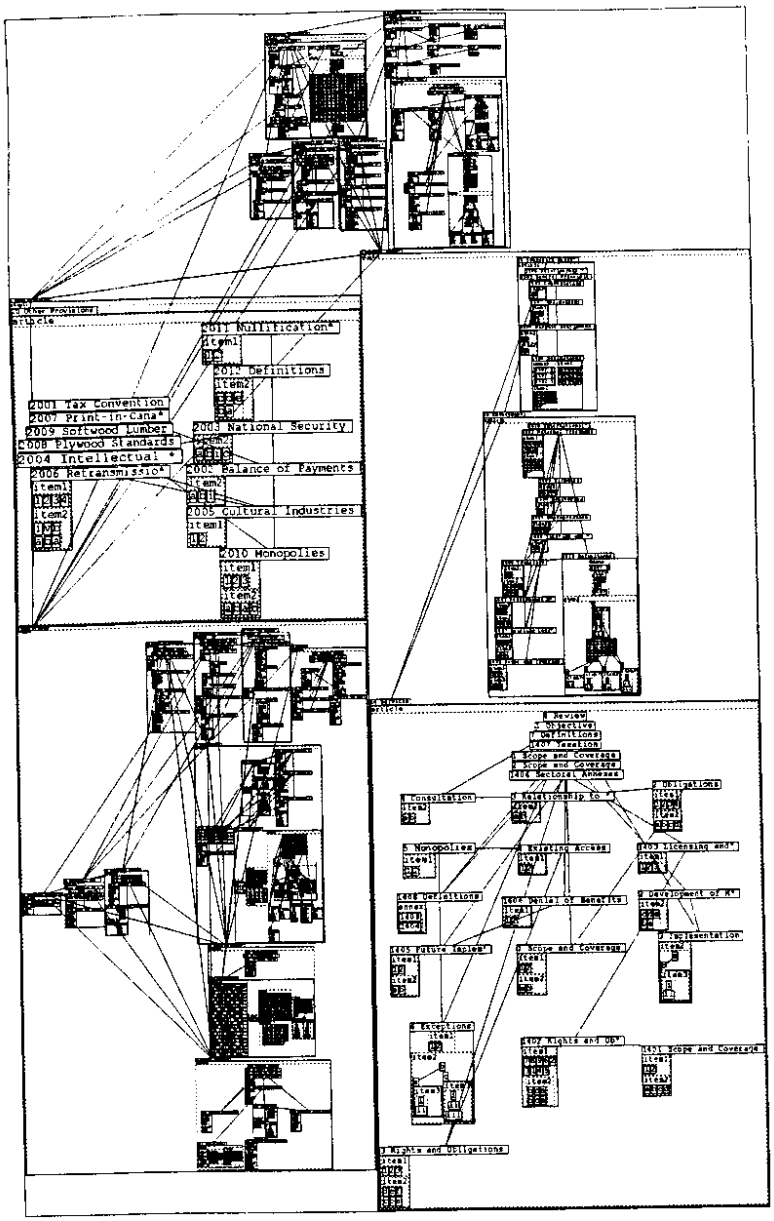

Figura 35: Layout de grafos aninhados utilizando vários pontos focais

3.6 Analisando Técnicas para Construir Diagramas

Neste capítulo, mencionamos algumas técnicas para a construção de diagramas legíveis e que tentam fornecer ao usuário uma visão global do espaço de informações.

O algoritmo de Sugiyama, apresentado na Seção 3.1, é a técnica clássica para layout de grafos e pode ser utilizada para a construção de browsers simples, sem subgrafos aninhados. Para a construção de diagramas com estruturas de grafos aninhadas, a técnica que deve ser utilizada é a descrita na Seção 3.3, que dá um tratamento diferenciado para relações de inclusão e adjacência entre nós. Esta técnica é bastante interessante para a construção dos browsers propostos nesta dissertação, pois representa claramente a estrutura de um modelo hipermídia com composições aninhadas.

A representação de estruturas de grafos hierárquicas através de conjuntos aninhados (vide Seção 3.2) é fundamental para a apresentação das relações de inclusão entre os nós, diferenciando-as das relações de adjacência, sendo utilizada inclusive na proposta apresentada na Seção 3.3. A herança de elos descrita na seção 3.3 minimiza a quantidade de elos exibida e melhora consideravelmente a legibilidade do diagrama, por outro lado, omite muitas informações que talvez sejam importantes para o usuário.

O conceito de HYPERDRAWERS, introduzido na Seção 3.4, é muito interessante, pois mostra como o agrupamento de nós relacionados pode melhorar a apresentação de uma estrutura, ajudando a orientação do usuário. Este conceito será aproveitado nesta dissertação, principalmente porque o agrupamento dos nós já é oferecido naturalmente pelos modelos com composições aninhadas.

A Seção 3.5 discute a técnica que será utilizada, com as devidas extensões introduzidas, para a construção dos browsers propostos neste trabalho. As visões olho-de-peixe possibilitam a construção de mapas que exibem a estrutura global de hiperdocumentos fornecendo também detalhes sobre informações locais relativas a posição do usuário na estrutura. Este é objetivo principal dos browsers aqui propostos: reunir informações globais e locais em um único diagrama de forma legível.

As extensões à proposta de Furnas descritas nas Seções 3.5.1 e 3.5.2 são interessantes, pois introduzem considerações de layout na apresentação de diagramas, utilizando a estratégia de olho-de-peixe para modificar o tamanho e a posição dos nós no mapa. Estas propostas diferem da extensão ao modelo olho-de-peixe a ser apresentada no próximo capítulo desta dissertação, pois não definem as componentes da função de grau de interesse do usuário em relação à distância abstrata entre os nós, considerando todos os componentes da estrutura de nós e elos. Na proposta da Seção 3.5.1, a distância entre nós corresponde a sua distância euclideana no layout inicial do grafo, não tendo nenhuma relação com sua estrutura. A proposta da Seção 3.5.2 só considera a hierarquia de composições para computar a distância, sem levar em conta os elos da estrutura. Estas propostas poderiam ser utilizadas para melhorar o layout das visões olho-de-peixe propostas por esta dissertação, mas não para filtragem de informações a serem exibidas para o usuário, que é o objetivo principal da dissertação ao utilizar a estratégia olho-de-peixe.

Para a construção dos browsers para documentos hipermídia com composições aninhadas, propõe-se que a função de grau de interesse seja definida considerando todos os aspectos estruturais dos hiperdocumentos. Esta proposta será apresentada detalhadamente no Capítulo 4.