UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Investigação de Técnicas de Visualização para Representação de Autômatos Finitos com Saída

por

DANIELA SATOMI SAITO

Dissertação submetida à avaliação,
como requisito parcial para a obtenção de grau de Mestre
em Ciência da Computação

Profª. Drª. Carla Dal Sasso Freitas

Orientadora

Porto Alegre, maio de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Saito, Daniela Satomi

Investigação de Técnicas de Visualização para Representação de Autômatos Finitos com Saída/ por Daniela Satomi Saito. – Porto Alegre: PPGC da UFRGS, 2003.

82 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, 2003. Orientador: Freitas, Carla Maria Dal Sasso.

1. Visualização de Informações. 2. Visualização de Grafos. 3.Autômatos Finitos com Saída. 4. Técnicas Foco+Contexto. 5. Agrupamento. I. Freitas, Carla Dal Sasso. II.Título.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais Celso e Anita pelo apoio dispendido durante todo o período deste mestrado. Estes foram os maiores incentivadores para a conclusão deste trabalho, dando apoio e muito amor para que eu não desistisse de atingir este objetivo.

À minha irmã Tatiana, e aos meus amigos, companheiros de todas as horas e que souberam compreender meus períodos de instabilidade e dúvidas durante o mestrado, e que me apoiaram mesmo assim.

À minha orientadora Carla Dal Sasso Freitas, pela orientação neste trabalho, sua dedicação e esforço em contribuir para o progresso do desenvolvimento das pesquisas da dissertação. Muito obrigada por toda a paciência e compreensão mesmo com as dificuldades da distância e de tempo de dedicação de minha parte.

E finalmente, agradeço a Deus, que sempre me iluminou, deu forças e protegeu durante todo este processo.

Sumário

Lista de Abreviaturas ou Siglas

2D	Bidimensional
3D	Tridimensional
2-conectado	Biconectado
API	A Priori Importance
DAG	Direct Acyclic Graph
CGI	Common Gateway Interface
DOI	Degree of Interest
DTD	Document Type Definition
EG	Edge Connectivity
ER	Entidade-Relacionamento
GD	Grafo Dirigido
GET	Graph Editor Toolkit
GLT	Graph Layout Toolkit
GML	Graph Modeling Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
LOD	Level of Detail
SHRimP	Simple Hierarchical Multiple Perspective
VC	Vertex Connectivity
WWW	World Wide Web
XML	Extensible Markup Language

Lista de Figuras

Resumo

Atualmente, a World Wide Web (WWW) já se estabeleceu como um dos meios de divulgação mais difundidos. Sendo um meio de publicação de custo relativamente baixo, muitas iniciativas foram desenvolvidas no sentido de estendê-la e transformá-la também numa ferramenta de apoio. Assim, uma série de pesquisas foi realizada no sentido de promover e facilitar o gerenciamento das informações da WWW, que são estruturadas, em sua maioria, como conjuntos de documentos inter-relacionados.

Grafos são estruturas utilizadas para a representação de objetos e seus múltiplos relacionamentos. Nesse sentido, pode-se afirmar que hiperdocumentos podem ser modelados através de grafos, onde uma página representa um nodo e um link para outra página é representado por uma aresta. Considerando estas características, e dada a crescente complexidade dos materiais publicados na WWW, desenvolveu-se, ao longo da última década, o uso de técnicas e recursos de Visualização de Grafos com larga aplicação na visualização da estrutura e da navegação na WWW. Técnicas de visualização de grafos são aplicáveis especificamente para representar visualmente estruturas que possam ser modeladas por meio de objetos relacionados, sendo investigadas técnicas para a abstração de modo a facilitar tanto o processo de compreensão do contexto da informação, quanto a apreensão dos dados relacionados.

Este trabalho tem como objetivo a investigação de técnicas de Visualização de Grafos aplicadas a autômatos finitos com saída. Este direcionamento se deve ao fato de alguns autores utilizar a abordagem de autômatos finitos com saída para as estruturas de hiperdocumentos. Se for considerado que um documento da WWW (ou o estado de um autômato) é composto por fragmentos de informação (ou saídas) tais como trechos de texto, imagens, animações, etc e que este documento é relacionado a outros por meio de links (ou transições), tem-se a verificação de sua representatividade por meio destas estruturas. Em trabalho anterior, no âmbito do PPGC da UFRGS, a ferramenta Hyper-Automaton foi desenvolvida com o objetivo de estender o uso da Internet no sentido de prover uma ferramenta de apoio à publicação de materiais instrucionais. Por adotar a notação de autômatos finitos com saída, possibilita, além da criação e gerenciamento de hiperdocumentos, a reutilização de fragmentos de informação sem que haja qualquer interferência de um autômato que utilize este fragmento sobre outro. O Hyper-Automaton foi selecionado como caso de estudo motivador deste trabalho. As técnicas aqui desenvolvidas têm como intuito diminuir a complexidade visual da informação, assim como permitir a navegação através dos autômatos finitos com saída de forma que seja possível visualizar detalhes como as saídas e informações relacionadas a cada uma delas, mantendo a visualização do contexto da informação. Foram analisadas técnicas de agrupamento como forma de redução da complexidade visual, e técnicas do tipo foco+contexto, como alternativa para prover a visualização simultânea do contexto e dos detalhes da informação.

Palavras-Chave: Visualização de Informações, Visualização de Grafos, Autômatos Finitos com Saída, Técnicas Foco+Contexto, Agrupamento.

TITLE:"VISUALIZATION TECHNIQUES FOR VISUALIZING FINITE AUTOMATAS WITH OUTPUT "

Abstract

Currently the World Wide Web (WWW) has already been established as one of the most popular source of information. Being a low cost publishing media, many initiatives were developed aiming at extending and transforming it as a support tool for many activities that rely on information usage. So, many works were developed to provide and facilitate the management of WWW information.

Graphs are structures used to represent objects and their multiple relationships. We can say that hyperdocuments can be modeled as graphs, where a page represents a node and a link to another page is represented by an edge. Due to these characteristics and the increasing complexity of the documents published on the WWW in the last decade, there has been intensive work on using techniques and resources of graph visualization in the visualization and navigation of the WWW. Graph visualization techniques are suitable to visually represent structures that can be modeled as a set of related objects. It is based on abstraction techniques to facilitate the process of comprehension of context information and the interpretation of information.

This work investigates graph visualization techniques for the visualization of finite automata with output due to the fact that some authors represent hyperdocuments structures as finite automata with output. This approach considers a WWW document (or a state) as composed by information fragments (or outputs) as texts, images or animations, related to others by transitions (or edges).

In a previous work at the PPGC/UFRGS, the software called Hyper-Automaton was developed, with the goal of extending the use of WWW by providing a tool to support the publication of instructional material on the Web. By adopting the notation of finite automata with output, it makes possible the creation and management of hyperdocuments, as well as the reuse of information fragments by independent automata. Hyper-Automaton was selected as the case study in our work. The techniques developed here aims at providing support for navigation through the automaton, while diminishing the visual complexity in information display, but allowing the visualization of details, like the outputs and their related information, and the context. Clustering techniques were analysed as a solution to reduce the visual complexity, and focus+context techniques as an alternative to provide the simultaneous visualization of detailed and contextual information.

Keywords: Information Visualization, Graph Visualization, Clustering, Finite Automata with Output.

1 Introdução

A Visualização de Informações é uma área de pesquisa e desenvolvimento relativamente nova na Ciência da Computação. Tem como objetivo o estudo de técnicas que possam prover meios de representação visual de dados, normalmente abstratos ou complexos demais, para uma melhor compreensão dos mesmos [GER 97]. Geralmente esta tarefa é realizada através de abstrações do mundo real que estimulem a percepção humana. Tais abstrações visuais podem ser manipuladas interativamente de modo a facilitar ainda mais a percepção. Deseja-se, desta forma, prover uma maior compreensão da informação visualizada.

O argumento padrão utilizado nesta área é que a exploração do processamento visual pode ajudar as pessoas na compreensão dos dados ou em outras tarefas relacionadas. Isso se deve ao fato de que no cérebro, 70% de todos os receptores e mais de 40% do córtex é destinado à visão. Ou seja, a tentativa de localização de padrões dentro de dados alfanuméricos requer maiores inferências da memória do que a mesma tarefa realizada com auxílio de recursos visuais [MUN 2000].

A descoberta das informações através da visualização é um processo que se inicia normalmente com as tentativas de visualização de alguns aspectos de todo um conjunto de dados para que assim, seja possível descobrir como proceder uma investigação sobre estes dados. Trata-se de um campo ativo de estudos, já que existem muitas questões abertas ou ainda não compreendidas no que diz respeito ao *design*. De qualquer forma, seja qual for o tipo de visualização estudado, é geral o uso de elementos gráficos como uma forma de sintaxe visual para a representação do significado das informações (semântica).

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

A representação de relacionamentos leva ao conceito de grafo, já que um grafo representa dados que possuem relacionamentos entre si. Como exemplos de uso, podem ser citados modelos de sistemas de software, de mapas de sites, de sistemas de tempo real e autômatos, entre outros. Este conceito, intuitivamente, leva à noção de um diagrama de nodos conectados. Entretanto, há uma infinidade de formas de representar visualmente um grafo.

Nas aplicações da visualização de informações, a utilidade do desenho de um grafo depende da sua legibilidade, ou seja, da capacidade de transportar o significado do diagrama rápida e claramente. Problemas como o mau aproveitamento do espaço e navegação em grafos de tamanho grande, entre outros, prejudicam a legibilidade e se transformaram em tema constante de pesquisas da área. Como resultado destas

pesquisas surgiram várias técnicas de desenho de grafos, algoritmos e ferramentas, assim como mais e mais requerimentos de análise para a obtenção do grafo considerado ótimo.

Esta dissertação tem como meta investigar técnicas de visualização de grafos, com duas diretrizes principais:

- Permitir a visualização de estruturas como autômatos finitos com saída;
- Reduzir a complexidade visual da informação representada no grafo.

O trabalho guiado por estas diretrizes deverá permitir ao usuário a facilidade de apreensão da informação e a facilidade de compreensão das estruturas visualizadas. Ou seja, através da redução da complexidade visual dos grafos exibidos como autômatos finitos com saída, espera-se proporcionar a facilidade de compreensão das estruturas.

As técnicas investigadas neste trabalho foram aplicadas ao caso de estudo representado pelo sistema Hyper-Automaton [MAC 2000][MEN 2000], desenvolvido como dissertação de mestrado no PPGC do Instituto de Informática da Universidade Federal do Rio Grande do Sul. O Hyper-Automaton vem sendo utilizado para a construção, gerência e exibição de hiperdocumentos na Web.

No presente trabalho foram estudados os fundamentos de Visualização de Grafos, justamente devido à visão de sua aplicabilidade à visualização dos autômatos finitos com saída. Foram investigadas técnicas do tipo foco+contexto [CAR 99], visando proporcionar aos usuários uma visão geral do contexto das informações juntamente com o acesso aos detalhes de cada nodo ou saída. Já para a redução da complexidade visual da informação, foram estudadas técnicas de agrupamento, para simplificar a estrutura da informação, mas mantendo uma visão geral da estrutura.

O restante do texto está organizado em cinco capítulos. No capítulo 2 são apresentados os principais conceitos de grafos, que são a base para a execução deste trabalho: é feita uma breve descrição teórica sobre grafos, assim como são citadas algumas propriedades consideradas relevantes para o trabalho. Em seguida, no capítulo 3, apresenta-se uma revisão bibliográfica sobre técnicas de visualização de grafos, dando ênfase às técnicas foco+contexto (principalmente à visão olho-de-peixe [FUR 86]) e às técnicas de agrupamento, que são objeto de implementação nesta dissertação. O capítulo 4 faz uma descrição da ferramenta Hyper-Automaton, utilizada como estudo de caso. Logo após, no capítulo 5, são apresentados os estudos realizados sobre esta ferramenta e as técnicas de visualização propostas. E, finalmente, no capítulo 6 são apresentadas conclusões e sugestões de trabalhos futuros.

2 Breve Revisão sobre Grafos

Este capítulo tem como intuito apresentar as definições essenciais de grafos além de apresentar algumas propriedades relevantes que serão abordadas posteriormente. O capítulo foi escrito baseado, principalmente, no livro "Teoria dos Grafos" de James A. McHugh [MCH 90].

2.1 Conceitos

Grafos são objetos matemáticos que podem ser utilizados para modelar sistemas e estruturas onde o relacionamento entre os objetos tem um papel importante, senão essencial, para a obtenção do resultado final. São comumente utilizados em áreas como modelagem de redes, estruturas de dados, escalonamento de processos, diagramas entidade-relacionamento (ER), entre outros.

Pela definição matemática, um grafo G = (V, E) é um conjunto de vértices V conectados por arestas do conjunto E [AHO 76]. Comumente, numa representação gráfica, os vértices são representados por círculos ou caixas, e são conectados pelas arestas (Figura 2.1), que são as linhas que indicam a existência do relacionamento entre os objetos. Ou seja, a representação de um grafo, formalmente, é uma abstração das relações entre os elementos de dados num espaço Euclidiano de *n* dimensões através de um conjunto de vértices e curvas contínuas.

Além disso, um grafo será denominado rotulado em vértices (ou arestas) quando a cada vértice (ou aresta) estiver, respectivamente, associado um conjunto denominado rótulo. Assumindo que os nodos e arestas identificados nos conjuntos V e E da Figura 2.1 correspondem também aos rótulos pertencentes a cada um deles, tem-se como resultado um grafo rotulado tanto nos vértices como nas arestas.

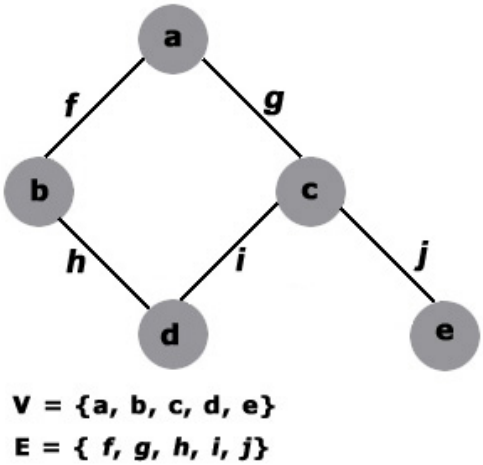


FIGURA 2.1 - Exemplo de um grafo

Pode-se dizer que um vértice *u* em V(G) é adjacente ao um vértice *v* em V(G) se a = (u, v) for uma aresta em E(G). Os vértices *u* e *v* são chamados de terminais da aresta (u, v), e a aresta *a* é dita incidente aos vértices *u* e *v*. Dado um conjunto de vértices S em G, define-se o conjunto de adjacência de S, denotado por ADJ(S), como o conjunto de

vérties adjacentes a algum vertice em S. Na Figura 2.2, por exemplo, o par de vértices {a, b} são os terminais da aresta *f*, que é incidente aos vértices *a* e *b*. Já o conjunto de vértices adjacentes ao nodo *a*, denotados por ADJ(a) são identificados como ADJ(a) = {b, c}.

Um vértice que não possua nenhuma aresta incidente é chamado de vértice isolado, enquanto que um par de arestas incidentes sobre um vértice comum são ditos adjacentes. Na Figura 2.2, o conjunto de arestas V possui dois vértices isolados, que são *d* e *e*; além disso, o conjunto de arestas E, composto pelas arestas *f* e *g* é adjacente, uma vez que *f* e *g* incidem sobre o vértice *a*.

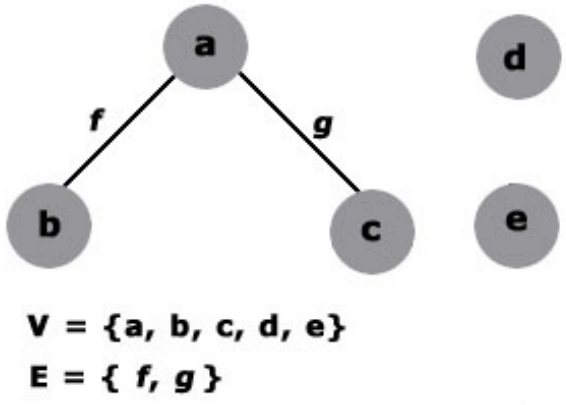
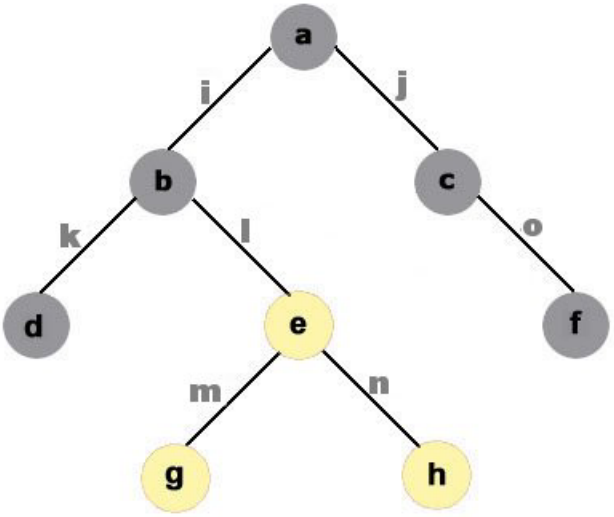


FIGURA 2.2 - Exemplo de um grafo com vértices isolados

Um par de vértices *u* e *v* num grafo é conectado se, e somente se, há um caminho de *u* a *v*. No entanto, um grafo G é dito conectado se, e somente se, todo par de vértices em G forem conectados. Por exemplo, na Figura 2.2, pode-se dizer que o par de vértices *a* e *b* está conectado, pois há um caminho *f* entre eles; no entanto, o grafo não é dito conectado porque podem ser encontrados elementos isolados nele.

Um sub-grafo S de um grafo G = (V, E) é um grafo S = (V', E') onde V' está contido em V, E' está contido em E, e os terminais ou pontos finais de qualquer aresta em E' também estejam em V'. Ou seja, um sub-grafo G' = (V', E') de um grafo G = (V, E) é um grafo tal que $V' \subseteq V$ e $E' \subseteq E$. Por exemplo, na Figura 2.3, dados os conjuntos V e E de G já identificados, o sub-grafo S destacado (em amarelo) é composto pelos conjuntos S' = {e, g, h} e E' = {6, 7}. Vale ressaltar que este não é o único sub-grafo contido em G, e sim, uma demonstração de um dos sub-grafos que podem ser identificados dentro deste conjunto. Por exemplo, se o conjunto V' identificado fosse igual a V, este sub-grafo poderia ser chamado de *spanning subgraph.*



V = { a, b, c, d, e , f, g, h }
E = {*i, j, k, l, m, n, o*}

FIGURA 2.3 - Exemplo de um sub-grafo

Caso sejam impostas direções sobre as arestas do grafo, as arestas serão interpretadas como pares de vértices ordenados, ao contrário das arestas não-dirigidas, que são consideradas como pares não-ordenados. Sendo assim, as estruturas correspondentes serão chamadas de grafos dirigidos (Figura 2.4) ou *digraphs*. Dado um par de vértices defindo por (*u, v*), o primeiro vértice *u* deste par é chamado de vértice inicial enquanto o vértice *v* é chamado de vértice terminal da aresta dirigida. Considerando que G seja um grafo dirigido, e que (*u, v*) seja uma aresta de G, tem-se que o vértice *u* é adjacente ao vértice *v*, sendo *v* o vértice terminal e adjacente a partir do nodo *u*.

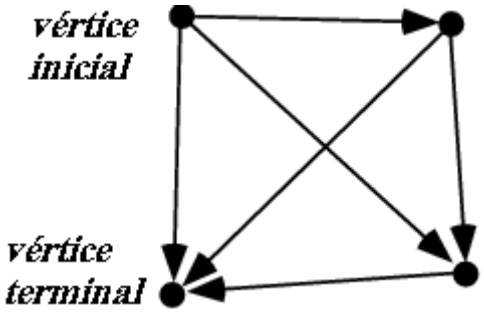


FIGURA 2.4 - Exemplo de grafo dirigido

Num grafo dirigido, o número de vértices adjacentes a um vértice *v* é denominado como grau de entrada de v, denotado como *indeg(v)*. O número de vértices adjacentes a partir de *v* é denominado grau de saída, que é denotado por *outdeg(v).*

Um grafo dirigido G(V, E) é chamado de fortemente conectado, se houver um caminho dirigido entre todos os pares de vértices em G. Um vértice *u* é dito atingível por um vértice *v* em G, se houver um caminho dirigido de *v* para *u* em G. O grafo dirigido obtido à partir de G adicionando a aresta (*v, u*) entre qualquer par de vértices *v*

e *u* em G a qualquer sempre que *u* for atingível à partir de *v* (e (*v, u*) ainda não está em E(G)) é chamado de fecho transitivo de G.

2.2 Tipos de Grafos

Além da diferenciação entre grafos dirigidos e não-dirigidos, existe uma série de categorias de grafos que podem ser enumeradas. Multigrafos, por exemplo, são grafos onde é permitida a existência de mais de uma aresta entre um par de vértices (Figura 2.5), multigrafos com arestas dirigidas são chamados de multigrafos dirigidos. Em contraste aos multigrafos, grafos que não permitem arestas paralelas, são chamados de grafos simples. E, finalmente, existem também aqueles em que existem tanto arestas dirigidas quanto não-dirigidas. Estes grafos podem ser enquadrados na categoria de grafos mistos.

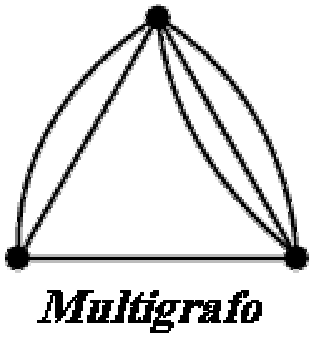


FIGURA 2.5 - Exemplo de multigrafo

Um grafo que contém ciclos é chamado de grafo cíclico (Figura 2.6). Em oposição, grafos que não contenham ciclos são chamados de grafos acíclicos. Um grafo dirigido que não contenha ciclos dirigidos é chamado de Grafo Dirigido Acíclico ou DAG (Figura 2.7). O grafo dirigido especial mais comumente encontrado é a árvore, que é definida como um grafo acíclico conectado.

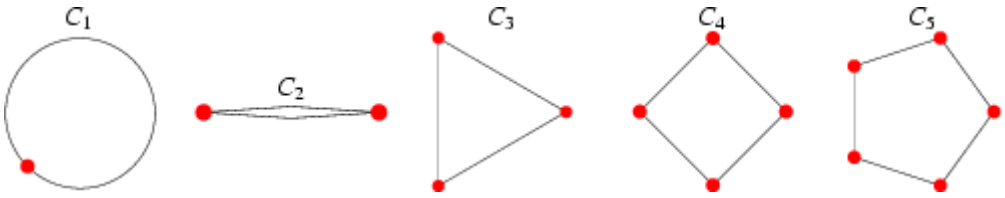


FIGURA 2.6 - Grafos não-dirigidos cíclicos

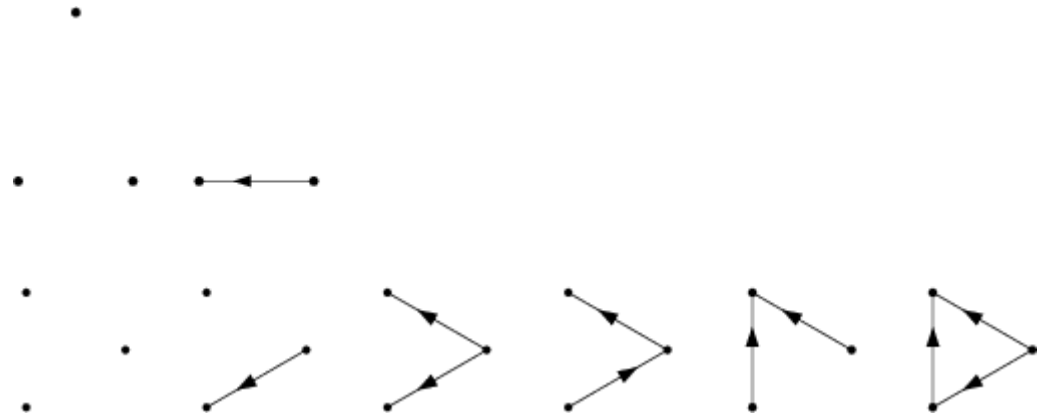


FIGURA 2.7 - Grafos dirigidos cíclicos

2.3 Conectividade

Dado um grafo G = (V, E), tem-se que a conectividade de vértice VC(G) é o número mínimo de nodos cuja deleção do grafo G o torne desconectado. A conectividade de aresta EC(G), por sua vez, trata-se do número mínimo de arestas cuja deleção do grafo G o transforme num grafo desconectado. As duas invariantes servem como medida de suscetibilidade de um grafo à desconexão sob a eliminação de um vértice ou aresta.

Um vértice cuja remoção aumenta o número de componentes num grafo é chamado de articulação (ou *cut-vertex*). Uma aresta cuja remoção faça a mesma coisa é chamada de ponte (ou *bridge*). Um grafo sem nenhuma articulação é chamado de não separável, biconectado ou 2-conectado. Um sub-grafo não-separável máximo de um grafo é chamado de bloco (componente biconectado ou bicomponente). Em geral, a conectividade do vértice (ou conectividade da aresta) de um grafo é o número mínimo de vértices cuja remoção resulte num grafo trivial desconectado. Chamamos um grafo G de *k-connected* ou *k-vertex connected (k-edge connected*) se a conectividade do vértice (ou aresta) for pelo menos k.

3 Visualização de Grafos

3.1 Introdução

A Visualização de Grafos se enquadra no contexto da Visualização de Informações, já que na representação de relacionamentos, como é o caso de grafos, temos dados abstratos, ou seja, dados que não necessariamente possuem uma geometria implícita.

A Visualização de Grafos congrega o desenvolvimento de técnicas que facilitem a percepção das informações contidas em estruturas cujos elementos encontram-se relacionados. As técnicas desenvolvidas têm como objetivo a espacialização de algum aspecto dos dados relacionados; tratam do layout de um grafo considerando algum atributo adicional que não seja apenas a informação das arestas. Estes atributos adicionais podem ser utilizados para influenciar outros atributos visuais da informação tais como cor e tamanho. A alteração destas características, por sua vez, pode ser empregada como técnica auxiliar na compreensão da estrutura de um grafo ou na realização de outras tarefas específicas tais como a seleção de um elemento ou apresentação do resultado de uma busca.

Existem diversos padrões de desenho de grafos, sendo que dentro de um determinado padrão, um grafo pode possuir ainda vários desenhos diferentes. Normalmente, os vértices são representados por símbolos como círculos, pontos ou caixas, e cada aresta entre pares de vértices (*u, v*), é representada por uma simples curva aberta entre os símbolos associados aos vértices *u* e *v*.

Na literatura, algumas características (ou aspectos) importantes no desenho de um grafo são ressaltadas. Duas delas são constantemente citadas [HER 99]:

- Previsibilidade – grafos iguais ou similares desenhados por um mesmo algoritmo não podem levar a representações muito diferentes. Isto é muito importante se o grafo mudar por alguma ação, por exemplo, escondendo (temporariamente) alguns nodos ou tornando-os visíveis novamente. Se, como resultado desta interação, o algoritmo de desenho de grafo criar uma visão muito diferente da anterior, poderá causar uma desorientação momentânea, diminuindo a usabilidade do sistema e fazendo com que o usuário gaste muito mais tempo na execução das tarefas relacionadas.

- Navegação em grafos grandes ou não-usuais – As aplicações práticas atuais levam à visualização de milhares, ou possivelmente dezenas de milhares de nodos. Para lidar com estes números, ferramentas de navegação, facilidades de busca, visões hierárquicas, etc. são necessárias. A implementação destas ferramentas também requer o uso de algoritmos de layout adequados.

Em conseqüência destes aspectos, deve-se tomar muito cuidado sobre o algoritmo de layout a ser escolhido. Existem vários algoritmos de layout de grafos que utilizam técnicas de otimização; deve-se analisá-los cuidadosamente, pois é inaceitável que pequenas alterações levem a representações visuais totalmente diferentes.

Mesmo assim, este fator não deve ser considerado como o maior problema: Herman *et al.* [HER 99] afirmam, em seu trabalho, que um ambiente de navegação rico é muito mais importante do que um bom layout. Este problema surge com muita ênfase no uso da *World Wide Web* (WWW). A expansão e a difusão do uso da WWW transformou o conceito de compartilhamento de informações. Esta mudança de conceitos fez com que houvesse o desenvolvimento de aplicações, produtos e serviços baseados neste repositório de informações. A WWW se popularizou e a hipermídia e os hipertextos vêm sendo cada vez mais utilizados como meio de publicação de materiais dos mais variados gêneros. Ou seja, são milhões de indivíduos que diariamente anotam e exploram todo este conteúdo de informações interligadas de maneira distribuída.

O conceito de hipertexto é antigo, sendo que o trabalho de Conklin [CON 87] pode ser citado como uma das primeiras fontes relacionadas ao seu desenvolvimento. A principal característica dos hipertextos é a possibilidade da divisão de um texto em unidades menores, formadas por conjuntos de páginas, representando uma determinada hierarquia como a de capítulos, seções, etc., com ligações entre elas, de modo que possibilite ao leitor a "navegação" pelo texto conforme a sua necessidade.

Sendo assim, pode-se afirmar que um hipertexto pode ser modelado utilizando a representação de um grafo dirigido. Num grafo dirigido G = (V, E), um conjunto finito de vértices V e um conjunto finito E de arestas, os pares (u,v) são ordenados. Num hipertexto representado por um grafo, as unidades do texto deverão corresponder aos vértices (ou nodos) e as conexões entre elas correspondem às arestas (ou ligações), de modo que a orientação das arestas será de acordo com o direcionamento de uma unidade de texto a outra.

Os hiperdocumentos surgiram para ampliar os potenciais oferecidos pelos hipertextos, utilizando elementos tais como imagens, sequências de animação, vídeos, trechos sonoros, entre outros. Na WWW, os hiperdocumentos são utilizados para as mais variadas aplicações, mas seguindo a estrutura de conexões de um grafo dirigido. Isso significa que a produção e gerenciamento destes materiais estão diretamente relacionados à criação e manipulação de grafos.

Como a exigência por velocidade na criação destas estruturas de informação é cada vez maior, houve um conseqüente aumento da complexidade das tarefas envolvidas. Com isso, ficou evidente a necessidade de métodos eficazes para a dinamização dos processos de criação e gerenciamento de hiperdocumentos. Sob este panorama, surgiram os primeiros estudos sobre a utilização da visualização de grafos como recurso auxiliar na execução destas tarefas. A geração de uma representação visual adequada, além de facilitar e acelerar o processo de cognição humana, facilita tarefas de gerenciamento de informações complexas, como a representação de cenários da Web. Diversas técnicas desenvolvidas ao longo dos últimos anos [MUK 95] [LAM 96] [MED 96] [MUN 98] foram utilizadas para a visualização de sites na Web, demonstrando, assim, a preocupação existente em prover uma visualização adequada desta base de informações. Deve-se ressaltar, entretanto, que essas técnicas são freqüentemente genéricas o suficiente para aplicação em outras áreas onde as entidades são modeladas como grafos.

3.2 Técnicas de Visualização de Grafos

3.2.1 Técnicas foco+contexto

A implementação do conceito de foco+contexto, segundo Card *et al.* [CAR 99], corresponde à criação de estruturas de informação de melhor custo, mostrando informações mais periféricas com detalhe reduzido combinadas com a informação no foco. Ou seja, trata-se de um conjunto de técnicas que permitem que o usuário dê foco a algum detalhe sem perder o contexto [HER 2000] da informação. Desta forma, quando uma região específica do grafo está com a atenção (ou foco) do usuário, ela é realçada em detrimento da apresentação das demais regiões, podendo ser alterada dinamicamente conforme a atenção do usuário seja alterada.

Há uma série de motivações para o desenvolvimento destas técnicas. Segundo Card *et al.* [CAR 99], alguns estudos anteriores como o de Larkin e Simon, de 1987, observaram apresentações que dividiam a informação em dois displays: um com o contexto e outro com o foco. Nestes estudos detectou-se uma considerável perda de performance devido ao uso dos dois displays. Como o espaço para as tarefas era limitado, a visualização de um volume de informação um pouco maior acabava implicando no uso de rolagem na visualização dos dados. Com isso, tarefas como as de localização das informações na janela de contexto sofriam prejuízos, acarretando um maior gasto de tempo para a execução das tarefas relacionadas.

Considerando estas dificuldades, foram levantadas três premissas básicas a serem sanadas nestas técnicas:

- necessidade de obter tanto a visão geral (contexto) da informação quanto à informação detalhada (foco) simultaneamente;
- possibilidade da diferenciação entre visão geral e visão que necessita ser detalhada;
- combinação destas informações dentro de um único display dinâmico, como na visão humana.

A primeira iniciativa encaixada deste gênero foi a visão olho-de-peixe, trabalho desenvolvido por Furnas [FUR 81], onde é simulado o efeito das lentes olho-de-peixe para a visualização das informações. Ou seja, a informação em foco é mostrada em maior detalhe ao mesmo tempo em que as informações restantes (ou o contexto da informação) são mostradas com uma quantidade de detalhes progressivamente menor, conforme o distanciamento do foco. Após este trabalho, uma série de novas propostas foi desenvolvida, sendo que várias delas são variações da visão olho-de-peixe.

Outra técnica bastante explorada nesta categoria é o layout hiperbólico. Esta técnica possui um efeito bastante similar ao da visão olho-de-peixe, porém, trabalha com o que se chama de geometria hiperbólica. Tem sido bastante utilizada em diversos trabalhos tendo como objetivo a visualização da WWW.

3.2.1.1 Visão olho-de-peixe

Desenvolvida por Furnas [FUR 81], a visão olho-de-peixe imita o efeito das lentes olho-de-peixe, que aumentam a visualização da área de maior interesse, no caso, o foco da visualização, e diminuem sucessivamente o detalhe das demais porções (Figura 3.1). Trabalhando desta forma, o usuário pode obter maior nível de detalhamento do foco escolhido sem perder o contexto da visualização, pois não deixa de visualizar a estrutura completa. É uma simples, mas poderosa técnica que complementa os mecanismos de *zoom* e *pan*. Mukherjea *et al.* [MUK 98] apresentaram uma adaptação desta técnica para a criação de um browser gráfico da WWW.

A visão olho-de-peixe requer uma estrutura dentro da qual as seguintes três propriedades podem ser definidas [FUR 81]:

- Ponto Focal – noção do ponto focal de interesse para interação com a estrutura.
- Distância do Foco – noção da distância de qualquer ponto da visualização do ponto de foco em questão.
- Nível de Detalhe (LOD – *Level of Detail*) – define noções como resolução, ou grau de detalhe, generalidade, etc.

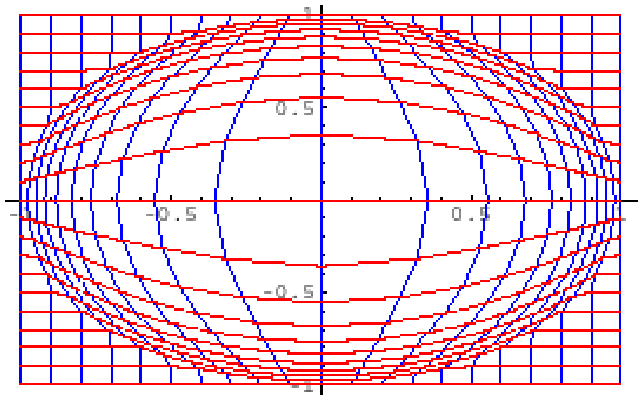


FIGURA 3.1 - Visão olho-de-peixe aplicada a uma grade [HER 2000]

Tendo em vista as propriedades descritas anteriormente, a função DOI – *Degree of Interest* – ou função de grau de interesse, pode ser definida. A função DOI é o mecanismo básico para prover a distorção da visualização. É ela a responsável por definir a importância de um nodo dentro da estrutura visualizada. Esta função tenta realizar um balanceamento entre o detalhe local (ou foco) e a estrutura global (ou contexto) de modo que o resultado visual seja satisfatório.

Por definição, a função DOI é composta pela soma de duas componentes. A primeira, é a importância a priori, ou API – *A Priori Importance* – que descreve a importância intrínseca de um nodo considerando toda a estrutura de um documento. Ela serve como parâmetro para a visualização dos dados, pois todos os outros nodos serão mostrados de acordo com o seu relacionamento com este valor. A segunda componente é a distância entre o nodo e o foco. A função DOI aumenta de valor com a importância a priori e diminui com a distância.

Noik [NOI 93] divide as técnicas de visão olho-de-peixe em duas categorias: as de filtro e as de distorção. As visões olho-de-peixe de filtragem utilizam um valor para restringir o display da informação aos elementos interessantes ou relevantes e são mais fiéis à concepção de Furnas. Já as técnicas de distorção trabalham com a idéia básica da distorção geométrica das visualizações para a obtenção do efeito das lentes olho-de-peixe. Ou seja, ao invés de filtrar os elementos, a distorção altera as posições, formas e tamanhos dos elementos.

A técnica desenvolvida por Sarkar e Brown [SAR 92] é uma delas; a visão é obtida por meio de transformações geométricas, considerando os pontos de foco e o tamanho dos nodos. Um aspecto interessante deste trabalho se refere à computação das transformações das arestas. Somente os seus pontos terminais são transformados como se pode ver na Figura 3.2, já que o mapeamento de um grande número de pontos pode se tornar impraticável para se obter um tempo de resposta real.

Já a técnica desenvolvida por Bartram *et al.* [BAR 95] apresenta uma abordagem diferente, onde as tranformações geométricas são realizadas através da alocação do espaço global a seções retangulares do display e baseando-se em fatores de escala seguidos de ajustes de grau de interesse (DOI). Schaffer *et al.* [SCH 98] utilizaram conceitos similares para o desenvolvimento de métodos de Full-Zoom.

Também foram desenvolvidos trabalhos no sentido de visualizar a estrutura com múltiplos pontos de foco, como Cava [CAV 2002] e Storey [STO 97], assim como Janececk e Pu [JAN 2000]. Estes últimos autores trabalham no sentido de gerar um framework para a implementação da visão de olho-de-peixe utilizando múltiplos contextos semânticos.

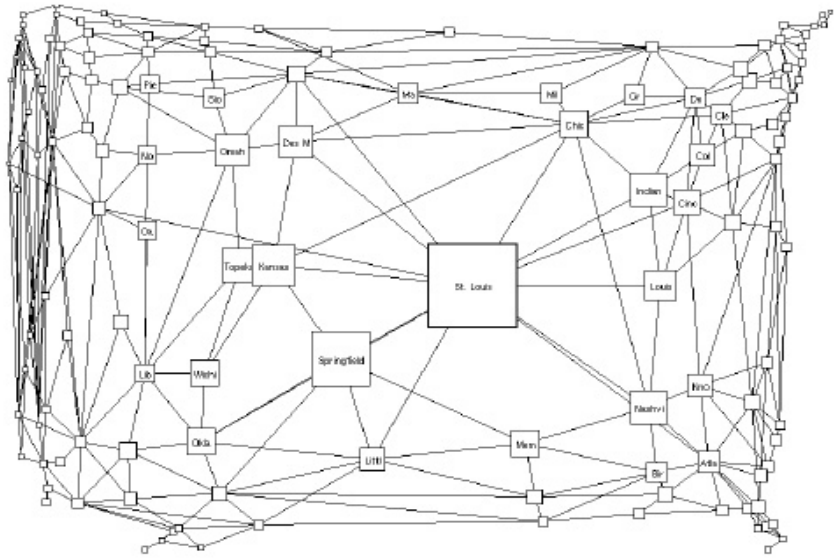


FIGURA 3.2 - Visão olho-de-peixe com a técnica de Sarkar e Brown [SAR 92]

O trabalho de Cava [CAV 2002], embora específico para estruturas hierárquicas e não para grafos, apresenta a utilização de dois focos na visualização de hierarquias ao invés de somente um, onde a área de detalhe mostra o nodo selecionado e sua sub-árvore, enquanto a área de contexto mostra o seu pai e suas sub-árvores, sendo que cada área possui seu próprio ponto de foco (Figura. 3.3).

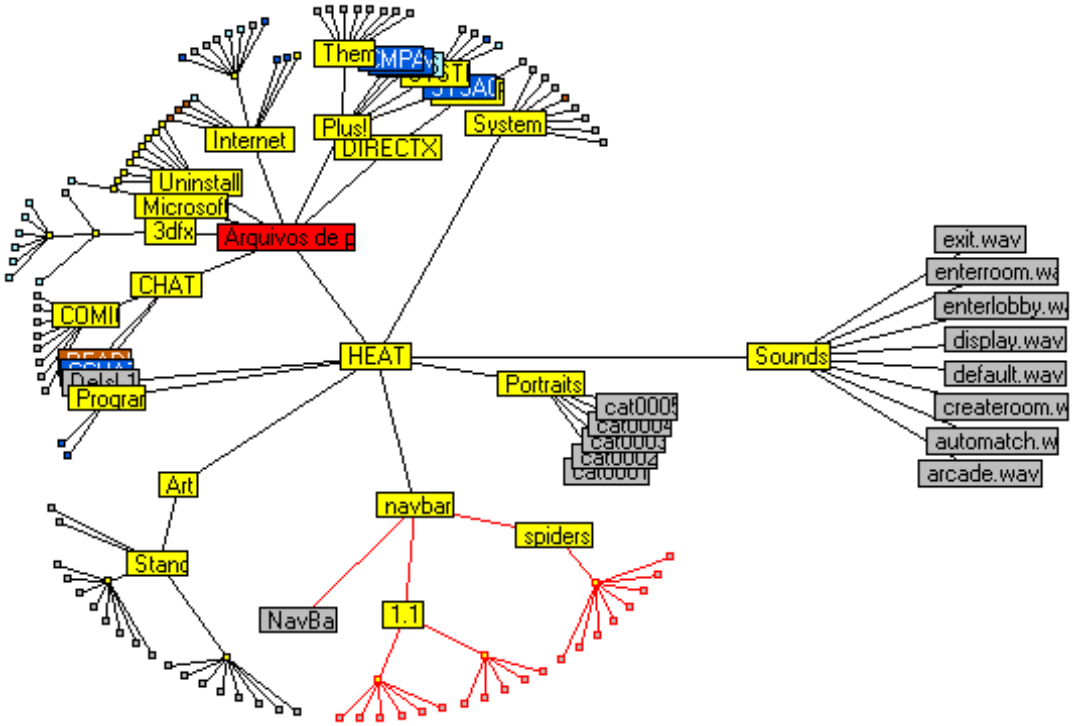


FIGURA 3.3 - Visão olho-de-peixe com a técnica de Cava [CAV 2002]

A técnica SHriMP – *Simple Hierarchical Multiple Perspective* [STO 97], por sua vez, tem como objetivo ajudar efetivamente estratégias híbridas de compreensão de programa. Esta técnica integra tanto as facilidades de *pan+zoom* como também a visualização olho-de-peixe para realizar a exploração da visão gráfica aninhada de estruturas de software. Gerencia múltiplos focos que podem ser necessários no exame de subsistemas e suas interconexões. A Figura 3.4 apresenta uma demonstração do uso da técnica.

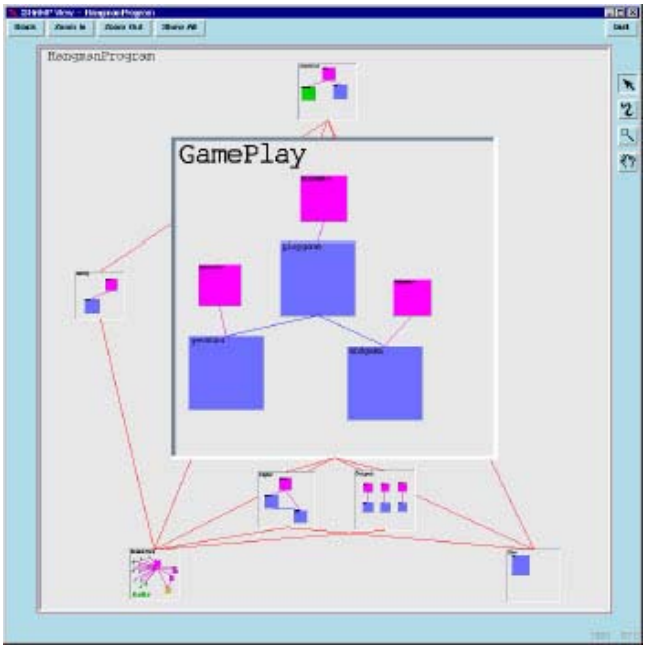


FIGURA 3.4 - Técnica de visualização SHriMP [STO 97]

Um dos pontos fracos da técnica olho-de-peixe de um modo geral, se refere à distorção aplicada aos grafos. Apesar da função de distorção na maioria das vezes ser relativamente simples, ela é matematicamente complexa. Segundo [HER 2000], a maioria dos sistemas gráficos usuais não oferece facilidades para a transformação destas linhas nas curvas distorcidas.

Como resultado, na maioria das vezes, opta-se por uma aproximação dos segmentos de linha originais com um alto número de pontos para se aproximar da curva transformada ideal. Além disso, para manter uma impressão suave, o número de pontos de aproximação deve ser relativamente alto levando a uma quantidade de cálculos muito grande, diminuindo a performance dos sistemas. Como foi visto anteriormente [SAR 92], houve uma proposta alternativa, onde apenas era realizada a alteração dos nodos de um grafo. A desvantagem, quanto a esta opção, é a geração de cruzamentos de arestas inexistentes na visualização inicial.

3.2.1.2 Layout hiperbólico

Mais uma técnica do tipo foco+contexto, o layout hiperbólico é aplicado principalmente em estruturas do tipo de árvores. É uma técnica de visualização desenvolvida visando a a possibilidade de interação com o usuário. Podendo ser implementado tanto em 2D como em 3D, este tipo de layout provê uma visão distorcida do layout da árvore, tornando possível a interação com árvores potencialmente grandes em aplicações reais.

Segundo Herman *et al.* [HER 2000], a geometria hiperbólica é baseada num sistema axiomático quase idêntico aos tradicionais axiomas Euclidianos. Herman *et al.* [HER 2000] afirmam que, no postulado Euclidiano, se uma linha não intercepta um ponto, então há somente uma linha paralela à linha original interceptando o ponto, enquanto que na geometria hiperbólica, há mais de uma linha paralela.

Pode-se dizer que os layouts hiperbólicos de grafos executam um algoritmo de layout no plano ou espaço hiperbólico, e então mostram seus resultados no plano ou espaço Euclidiano familiar utilizando um dos modelos de geometria hiperbólica. Ou seja, o que se vê não é a geometria hiperbólica por si, mas as suas representações na geometria Euclidiana. A própria natureza espacial diferente da geometria hiperbólica torna viável o uso de alguns algoritmos de layout. Alguns algoritmos de layout clássicos podem ser reutilizados num ambiente hiperbólico, mas levando a resultados relativamente diferentes.

O layout hiperbólico foi inicialmente proposto em 2D, com o *Hyperbolic Browser* de Lamping e Rao [LAM 95], que apresenta uma técnica para visualização e manipulação de grandes hierarquias. O objetivo da técnica é fazer o desenho da hierarquia de maneira uniforme no plano hiperbólico e mapeá-lo numa região circular (Fig. 3.5). O *Hyperbolic Browser* inicialmente mostra uma árvore com a sua raiz no centro, mas a visualização pode ser modificada conforme o ponto de foco é alterado. Em todos os casos, a quantidade de espaço disponível para um nodo é resultado de uma função da sua distância na árvore do ponto no centro.

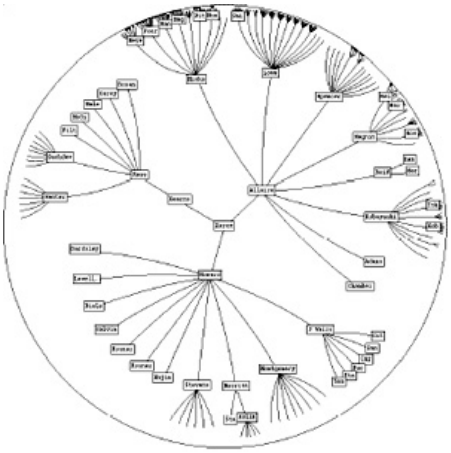


FIGURA 3.5 - Layout utilizando o Hyperbolic Browser [LAM 95]

O sistema H3Viewer [MUN 98] (Fig. 3.6), utiliza o layout de *cone trees* [CAR 93] dentro do espaço hiperbólico em 3D (Fig. 3.7) empregando o algoritmo de layout H3, descrito pela autora como o layout de "*cone tree* 3D hiperbólico de segunda geração". O algoritmo descrito distribui os nodos filhos na superfície de um hemisfério na base do cone ao invés de ao redor de uma circunferência. Num trabalho mais recente, o H3Viewer é empregado como base de um browser 3D e comparado a browsers 2D tradicionais [RIS 2000].

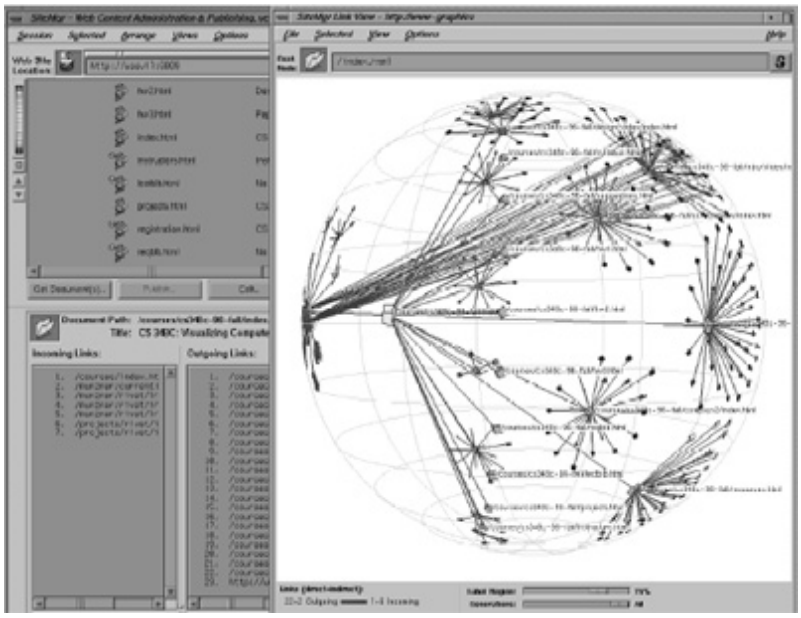


FIGURA 3.6 - O sistema H3Viewer [MUN 98]

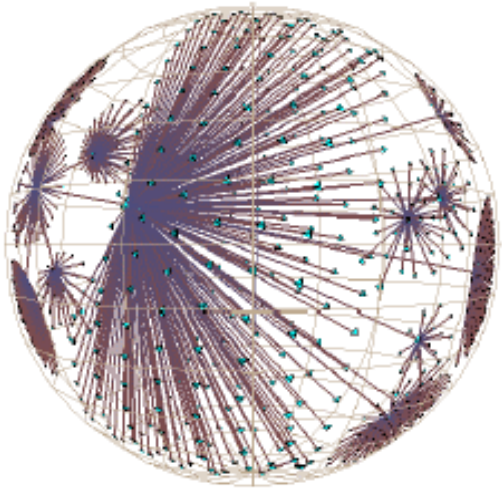


FIGURA 3.7 - Layout hiperbólico em 3D

3.2.2 Métodos de interação

Os métodos de interação são elementos essenciais na visualização de informações pois provêm operações que facilitam e auxiliam tanto a navegação quanto a visualização da informação geral. Estes métodos foram desenvolvidos justamente pelo fato de não existir nenhum algoritmo de layout que consiga resolver completamente todos os problemas decorrentes da visualização de grafos de tamanho grande nas aplicações de visualização.

Os tipos de interação criados até hoje para facilitar a visualização das informações podem se encaixar nas seguintes categorias [MUN 2000]:

- Navegação – a navegação interativa consiste da mudança do ponto de vista ou da posição de um objeto numa cena. Operações como as de *zoom*, *pan* e *scrolling* podem ser encaixadas nesta categoria.

- Escolhas – são comuns em ambientes não-navegacionais, por exemplo, através de botões num painel de controle ou escolhas de menu que afetem o display. Neste caso, as escolhas são implementadas para que possam ser executadas interativamente. Operações como a filtragem, a busca e a seleção pertencem a esta categoria, uma vez que nelas são selecionados elementos ou grupos de elementos para visualização ou interação.

- Transições animadas – o uso de animações para suavizar as transições entre as mudanças realizadas tem como intuito preservar o mapa mental que o usuário tem do objeto visualizado.

Esses tipos são brevemente descritos a seguir.

3.2.2.1 Zoom

Um dos mecanismos de interação mais conhecidos, o *zoom* é das operações tradicionais consideradas ferramentas praticamente indispensáveis, já que a maioria das aplicações de visualização de grafos atuais trabalha com uma quantidade de informações muito grande. North [NOR 94] e Gansner [GAN 98] descrevem em seus trabalhos o uso destes recursos em suas aplicações.

Os mecanismos de *zoom* podem ser classificados em *zoom* geométrico e *zoom* semântico. O *zoom* geométrico provê uma melhor visão do conteúdo do grafo, enquanto que o semântico muda o conteúdo das informações, apresentando mais detalhes quando se especifica uma área do grafo. O *zoom* geométrico é o que mais comumente é encontrado na maioria das aplicações, porém, em alguns casos, como o apresentado por Plaisant *et al.* [PLA 2002], o uso do *zoom* semântico mostra-se mais apropriado. Nesse mesmo trabalho, é citado o toolkit Jazz, desenvolvido por Bederson [BED 2000], que descreve o uso deste tipo de recurso. Com relação a esta categoria, deve-se dar atenção ao nível de detalhamento a ser atribuído a cada escala de visualização.

Todos os mecanismos descritos requerem que as mudanças sejam realizadas através de movimentos de transição suaves, pois mudanças muito bruscas podem provocar a perda do contexto da visualização por parte do usuário. Por exemplo, com o aumento da escala de *zoom*, a velocidade das mudanças fazem com que o ponto de foco desapareça tão rápido que o *pan* não consegue manter seu foco. Como resultado, o que ocorre é que ao invés da execução de um único passo, a operação se dará por meio de dois passos. Primeiro o *zoom* será executado, e somente depois é que o centro de visão será restabelecido através do *pan*.

É devido a estes problemas que técnicas alternativas, como as do tipo foco+contexto, foram desenvolvidas. Embora estas técnicas supostamente forneçam a visualização do contexto da informação e acelerem o processo de execução de tarefas para a maioria dos usuários, Schneiderman [SCH 98] afirma que para alguns, técnicas como as de full-zoom são mais apropriadas por apresentar maior detalhamento de

informações. Mesmo assim, a técnica foco+contexto apresentada no artigo, que é a de *variable zoom*, mostrou-se satisfatória para a maioria dos casos.

Dados os benefícios oferecidos pelas duas técnicas, uma série de novas técnicas foi desenvolvida agregando as funcionalidades de ambas. Dentre algumas encontradas na literatura, pode-se citar o *Flip Zooming* [HOL 97], que é utilizado para a visualização e download de documentos da Web e a técnica de *ShriMP* [STO 97], já mencionada. As técnicas integram tanto o *pan+zoom* como abordagens da visualização olho-de-peixe para explorar uma visão de um grafo aninhado da estrutura de software. A Figura 3.8 é um exemplo da técnica *Flip Zooming*.

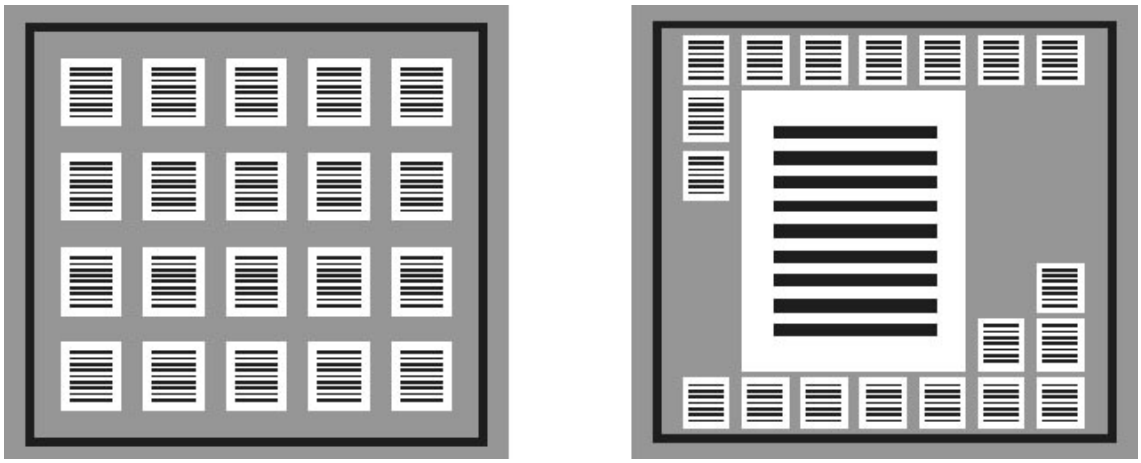


FIGURA 3.8 - Técnica de *Flip Zooming* [HOL 97]

3.2.2.2 Pan e scroll

Da mesma forma que o *zoom*, o *pan* e *scroll* podem ser encaixados na categoria das operações tradicionais mais conhecidas. Presentes também na maioria das ferramentas atuais, os mecanismos de *pan* e *scroll* podem ser descritos como métodos que provêm meios para o usuário se deslocar sobre a representação visual. Através deles, é possível selecionar diferentes trechos da informação para serem visualizados de maneira rápida.

O *scrolling* é realizado por um dos componentes de interface do usuário mais comuns, que são as *scroll bars* ou barras de rolagem [BJÖ 2001]. Elas permitem que os usuários naveguem rapidamente entre diferentes seções de um documento, além de indicar que seção do documento está na visão através da modificação do tamanho e posição do *scroller*. De maneira similar, o *panning* também permite que o usuário se desloque rapidamente através da informação visualizada.

As duas técnicas apresentam várias vantagens, mas mesmo assim, para Churcher [CHU 95], uma quantidade de aspectos referentes à interface com o usuário permanecem problemáticos. As dificuldades encontradas nas técnicas se devem justamente à simplicidade tão destacada nas mesmas. Assim como é possível percorrer o espaço da informação com facilidade, falta ao usuário uma quantidade maior de

informações à respeito do conteúdo real do espaço [MCC 99]. Outra dificuldade presente tem relação com os níveis de granularidade referentes às técnicas.

Alguns trabalhos [MCC 99] [BJÖ 2001] apresentam técnicas alternativas para sanar estas deficiências, ainda utilizando os conceitos de *scroll bars*. McCrickard e Catrambone [MCC 99] apresentam duas técnicas alternativas: o *Mural Bar* (Figura 3.9) e o *Pile Bar* (Figura 3.10). A técnica de *mural bar* provê uma visão geral de todo o espaço de informação mapeando os elementos do espaço para uma área menor, que é o espaço da *scroll bar* melhorada. Através desta *scroll bar* é possível realizar a navegação através dos dados, tendo uma visão geral do espaço de informação como um todo.

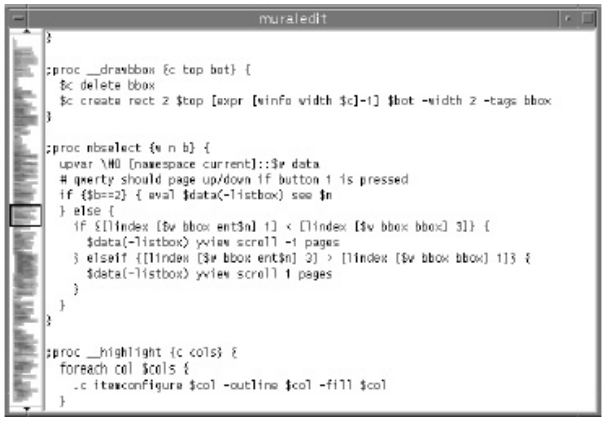


FIGURA 3.9 - Mural bar [MCC 99]

Já a técnica *Pile bar* introduz o conceito de empilhamento de itens (Fig. 3.10). Esta técnica requer que a informação a ser empilhada seja delineável como objetos distintos. Possíveis objetos a serem delineados incluem arquivos de código, entidades gráficas de uma linguagem de programação visual, ou entradas individuais de uma lista de texto. Dada uma divisão de um espaço de informação em objetos, é necessário criar representações de objeto de tamanho reduzido que caibam no espaço gráfico.

FIGURA 3.10 - Pile bar [MCC 99]

Björk [BJÖ 2001], por sua vez, desenvolveu a técnica *ScrollSearcher* (Fig. 3.11) para que os usuários tivessem um método integrado de busca de trechos de texto em documentos e pudessem explorar os resultados das buscas utilizando a manipulação direta através do estilo de navegação provido pelas *scroll bars*. Esta técnica acopla fortemente a apresentação dos resultados com a navegação provida pelas *scroll bars*. O estilo de navegação permite que os usuários se movam entre diferentes resultados

utilizando uma técnica de manipulação direta, além de permitir que os usuários escolham em que ordem ver estes resultados.

FIGURA 3.11 - Técnica *ScrollSearcher* [BJÖ 2001]

3.2.2.3 Filtragem de informação

A visualização de grandes quantidades de informações é um dos grandes problemas existentes na visualização de grafos, uma vez que a maioria das aplicações atuais lida com grandes quantidades de dados. Como resultado, a visualização desta quantidade de informações pode se tornar indesejável devido a razões como:

- a exibição dos pontos de informações apresentar sérios problemas de oclusão, ou seja, alguns objetos se sobrepondo a outros;
- haver uma quantidade de pontos de informação que o sistema não possa exibir em um tempo razoável;
- dados com muitas dimensões, o que torna impraticável a sua exibição em uma representação bidimensional ou tridimensional;
- interesse do usuário somente em um subconjunto particular de dados.

Para solucionar estes problemas, foram desenvolvidos mecanismos de interação como a seleção, filtragem ou refinamento de consulta, agrupamento e poda. Diferentemente dos mecanismos foco+contexo, que tentam conservar todas as informações contextuais, mesmo quando se trata de uma imagem muito pequena, estes mecanismos de interação permitem que o usuário selecione as informações a serem mostradas. O conceito básico aplicado nestes mecanismos é a diminuição do escopo da informação a ser visualizada e conseqüente diminuição da complexidade visual da estrutura informação, promovendo assim, uma melhor compreensão do contexto por parte do usuário.

Em termos gerais, a filtragem de informação realiza um processo de busca, onde as informações não relevantes são rejeitadas e os dados relevantes são apresentados de acordo com uma função ou parâmetro de interesse do usuário. Pode ser feita manualmente ou por meio de buscas dinâmicas, onde o usuário pode formular as buscas e obter controle sobre ela. Numa busca, por exemplo, são procurados e mostrados apenas os nodos que possuam a característica pela qual o usuário está procurando. Gansner [GAN 2000] e North [NOR 94] descrevem o uso de filtros em grafos, sendo que o trabalho de North [NOR 94] aplica-se a grafos de tamanho grande, complexos e difíceis de ler.

A filtragem de dados pode ser baseada na estrutura ou no conteúdo de um grafo, sendo que ainda pode haver uma combinação dos dois tipos. Seja qual for a escolha, as duas técnicas provêm um mesmo resultado: diminuição da quantidade de dados exibidos ao usuário sem causar nenhuma perda de informação relevante.

3.2.2.4 Busca e seleção

As operações de busca e seleção também podem funcionar com o objetivo implícito de reduzir o conjunto de dados a serem visualizados. Para tal, utilizam-se de recursos como os filtros de informação, os quais através de métricas estruturais e/ou de conteúdo realizam a tarefa desejada.

Num mecanismo de busca, um usuário do sistema de visualização realiza uma busca estabelecendo determinada característica desejada. A resposta desta operação age diretamente sobre a representação visual do conjunto de informações. Técnicas de visualização como a *ConeTree* [ROB 91], *Fisheye View* [FUR 81] e *Hyperbolic Browser* [LAM 95] ou ferramentas como as do pacote Graphviz, citadas por Gansner e North [GAN 2000] e Graphlet [HIM 96] fazem uso destas operações para prover maior interação com os usuários. Nas *ConeTrees* por exemplo, pode-se realizar uma operação de busca através do estabelecimento de parâmetros de consulta. Como resultado, é indicado o nodo que possui o maior escore para os parâmetros estabelecidos.

Já na seleção, o usuário tem a capacidade de selecionar, dentro de um conjunto de dados, uma determinada informação. A vantagem deste mecanismo é que ele permite que o dado selecionado seja visto com maior clareza, sem a sobrecarga visual embutida quando se visualiza o mesmo dado dentro da estrutura geral. Os sistemas SequoiaView [WIJ 99], que implementa o *Cushion Treemaps*, e o H3Viewer [MUN 98] provêm como uma de suas formas de interação a seleção de elementos, onde o elemento selecionado é destacado. Neles, é possível selecionar elementos com uma faixa de tamanho definida pelo usuário ou através da comparação de seu nome com expressões regulares.

O programa *lefty*, descrito por North [NOR 94] e Gansner e North [GAN 2000], também relata o uso destes recursos. Cohen *et al.* [COH 95] e Ryal *et al.* [RYA 96] também implementam estas técnicas, sendo que Cohen *et al.* [COH 95] apresentam métodos de busca e seleção como operações auxiliares para encontrar os nodos de um caminho ou de uma árvore.

3.2.2.5 Animação

Assim como toda técnica de visualização desenvolvida até hoje, os recursos de *zoom*, *pan* e *scroll*, além solucionarem alguns problemas de visualização e agregarem facilidades, trazem consigo alguns novos problemas, como já foi mencionado anteriormente.

A fim de evitar e solucionar os problemas de perda de contexto, introduziu-se o uso de animações na visualização das informações. Acredita-se que o uso deste recurso ajuda os usuários a manterem a consistência da visualização, além de facilitar o estabelecimento de relações entre os estados do sistema. Segundo Bederson e Boltman [BED 99], alguns pesquisadores demonstraram através de estudos informais que a animação pode melhorar a satisfação dos usuários dos sistemas. No entanto, apenas recentemente surgiram estudos no que se refere ao seu efeito sobre a performance do usuário [HOR 2001].

Sendo assim, pode-se definir como animação qualquer movimentação de objeto, ou qualquer alteração de sua aparência, aplicável tanto sobre os dados atualmente visualizados na interface, como alterações efetuadas sobre a própria interface com o usuário. No entanto, ao incluir métodos como estes, deve-se tomar muito cuidado. Como o objetivo proposto é facilitar a compreensão dos diferentes estados do sistema que está sendo visualizado, o tempo a ser empregado em cada transição também se transforma em um fator de extrema importância. Transições muito rápidas podem não obter os resultados desejados por não conferir tempo necessário para que os usuários do sistema se atentem às alterações empregadas e façam a conexão das informações que estão sendo visualizadas com as anteriores. Da mesma forma, transições longas demais também podem ser prejudiciais, por ”desconectar” o usuário da tarefa atual. Dados mostram que a satisfação do usuário de um sistema diminui conforme o tempo de atraso aumenta [BED 99]. Desta forma, o tempo de animação a ser empregado deve ser calculado levando em conta fatores como o tipo de tarefa e a experiência dos usuários com o tipo de interface que está sendo utilizada.

Considerando os méritos de desempenho do sistema, não há como negar que o uso de animações implica no aumento do tempo de resposta total do sistema. No entanto, deve-se fazer uma avaliação de qual aspecto será mais vantajoso: a utilização de um sistema mais lento, que demonstra as alterações e transições da informação através de animações que mantêm o usuário focado na informação desejada; ou um sistema de resposta imediata, onde o usuário gasta tempo para a realização de ajustes para a compreensão da nova visualização apresentada.

Para exemplificar, alguns sistemas que fazem uso de recursos de animação são descritos nos trabalhos de North [NOR 94], Baker *et al.* [BAK 96], Herman *et al.* [HER 99] e Lisistyn e Kasnayov [LIS 99]. O sistema Latour, por exemplo, descrito por Herman *et al.* [HER 99], apresenta todas as mudanças possíveis por meio de animações visando evitar ao máximo qualquer mudança radical que prejudique o usuário. O autor também faz o uso dessa técnica para reduzir possíveis problemas de usabililidade.

3.3 Técnicas de agrupamento em grafos

Os processos de agrupamento têm como fundamento a identificação de conjuntos de padrões e classes dentro de um conjunto de dados. Para que se possam realizar estas identificações, é necessário que, inicialmente, exista uma métrica definindo os pontos de similaridade ou padrões a serem detectados.

No caso de agrupamento em grafos, com a utilização de tais métricas, os nodos serão reunidos de forma que os padrões detectados acarretem a divisão dos grafos em conjuntos de nodos disjuntos: nodos pertencentes a um mesmo agrupamento possuem comportamentos ou características similares e nodos pertencentes a agrupamentos diferentes diferem com relação a estes quesitos.

A detecção dos padrões de agrupamento de um grafo pode se dar tanto baseada nas informações estruturais de um grafo (chamada às vezes de agrupamento natural, ou particionamento), como nas propriedades semânticas do domínio da aplicação, também chamado de agrupamento por conteúdo. Métricas para definição de agrupamentos serão tratadas na seção 3.3.1.

Pode-se dizer que um dos principais objetivos da aplicação da técnica de agrupamento diz respeito à redução da complexidade visual. Porém pode-se encontrar na literatura referências à sua utilização em tarefas como análise de estrutura ou com operações como busca e seleção.

A redução da complexidade visual tende a garantir maior velocidade e efetividade nos processos de visualização e compreensão dos dados apresentados e propicia melhora na performance do sistema. Estes impactos são mais perceptíveis em sistemas que lidam com grandes quantidades de informação. Através da seleção de critérios de agrupamentos adequados, é possível destacar uma série de características que, por muitas vezes, passam desapercebidas aos olhos dos usuários. Isto acontece principalmente devido ao excesso de elementos apresentados simultaneamente, ou mesmo devido à própria natureza da informação.

Nas operações de filtragem e busca, o agrupamento é utilizado como um método complementar pelo fato de poder ser implementado como uma função numérica computável, sendo utilizada, então, como critério para as duas operações. Em termos de visualização, a filtragem se refere à desenfatização ou remoção de elementos da visão, enquanto que a busca normalmente se refere à ênfase de um elemento ou grupo de elementos. Sendo assim, as técnicas de agrupamento são devidamente aplicadas para que grupos de elementos sejam selecionados e devidamente destacados, no caso do busca, e que elementos sejam escondidos ou removidos da visualização, no caso da filtragem.

3.3.1 Métricas

Métricas podem ser definidas como medidas numéricas associadas aos nodos e que são utilizadas como parâmetro para a realização do agrupamento dos nodos de um grafo. Podem ser utilizadas tanto para medir quanto para quantificar algum aspecto mais abstrato de um nodo, permitindo assim a comparação com outros nodos do mesmo tipo.

Herman *et al.* [HER 2000] referem-se ao termo métrica como sendo uma medida que é associada, ou relacionada, aos nodos de um grafo.

As métricas podem ser utilizadas para uma variedade de propósitos diferentes, e, na visão desses autores, todas as aplicações possíveis ainda não foram completamente exploradas. Um exemplo de utilização destas métricas é encontrado no processo de extração de *spanning trees* de grafos.

Por poderem ser implementadas como uma função numérica computável, as métricas podem ser utilizadas ainda na implementação de outros mecanismos de interação como a busca e a filtragem. Técnicas como estas utilizam os valores das métricas como critério de apresentação visual dos resultados. Por exemplo, numa filtragem, apenas as informações que possuem um determinado valor medido é que são destacadas para a visualização.

Basicamente, existem as métricas computadas pela estrutura ou pelo conteúdo, sendo que ainda pode haver uma métrica baseada em estrutura e conteúdo. As métricas baseadas em estrutura utilizam como critério apenas informações referentes à estrutura do grafo, ou seja, quantidade de nodos, arestas, ciclos e quaisquer outras informações estruturais.

Um exemplo deste tipo de métrica é o grau de um nodo, ou seja, o número de arestas conectadas a este nodo. Para que os nodos que possuem um maior número de relações com os outros possam ser visualizados, basta utilizar esta métrica para evidenciar apenas os nodos com um grau maior ou igual a um determinado valor estabelecido. Outro bastante comum é a localização de grupos disjuntos ou mutuamente exclusivos. Os grupos disjuntos são selecionados por serem mais simples de navegar; no entanto, nem sempre é possível a localização destes grupos. Em Marshall [MAR 2000], por exemplo, é descrito o uso da técnica *Skeleton Extraction*, que permite que o usuário obtenha a visão do esqueleto do grafo. Este resultado é obtido porque existe um valor de métrica associado aos nodos, sendo extraídos os nodos com valor de métrica abaixo do valor de corte.

Já as métricas baseadas em conteúdo, baseiam-se em informações associadas aos nodos como, por exemplo, um texto ou qualquer outra informação adicional. Ou seja, proporciona a utilização de métricas de agrupamento com maior nível de exatidão. Para que possa ser aplicada, deve haver um certo conhecimento sobre o domínio da informação da aplicação.

É fato que a métrica estrutural pode ser aplicada a qualquer grafo independente de qualquer conhecimento sobre o domínio da informação. Entretanto, a utilização da métrica baseada em conteúdo pode apresentar resultados mais específicos, dependendo da necessidade da aplicação. Mesmo assim, a maioria das referências encontradas na literatura trata do agrupamento estrutual. Talvez isso se deva à necessidade de conhecimento da informação por parte dos usuários para a utilização da técnica baseada em conteúdo, assim como a dificuldade de generalização da técnica desenvolvida para outras aplicações. Uma vantagem da técnica estrutural seria o fato de que, devido às métricas que utiliza, mantém a estrutura do grafo agrupado segundo um sub-conjunto dos relacionamentos originais, o que pode ajudar na orientação do usuário, além de permitir a generalização para qualquer aplicação.

Apesar desta diferenciação entre as métricas, ainda é possível realizar uma combinação da aplicação das duas métricas, formando uma métrica composta, com o intuito da obtenção de efeitos mais poderosos. Um exemplo simples é permitir que um usuário adicione um valor de "peso" específico da aplicação dos nodos. Desta forma, os valores de peso são utilizados juntamente com as métricas estruturais para um resultado de maior exatidão.

3.3.2 Técnicas de agrupamento

Existe, na literatura, uma série de técnicas ou métricas de agrupamento visando atender os mais variados objetivos sendo que elas podem ser classificadas de acordo com vários critérios. Tölle e Niggemann [TÖL 2000] citam como exemplo de critério as características dos algoritmos utilizados (algoritmos hierárquicos ou não-hierárquicos). Outro critério seria o das características dos métodos. Por exemplo, um método pode ser exclusivo ou não. Num método exclusivo, todo nodo é associado a exatamente um agrupamento, enquanto que um método não-exclusivo pode atribuir um nodo a vários agrupamentos.

3.3.2.1 Agrupamentos Hierárquicos e Não-Hierárquicos

No agrupamento hierárquico, uma hierarquia de agrupamentos é construída, formando uma árvore onde cada agrupamento é composto por outros grupos de tamanho menor [BEK 02]. Este tipo de agrupamento é aplicado recursivamente sobre as estruturas. Os métodos de agrupamento podem proceder de uma forma divisiva (abordagem *top-down*), ou de forma aglomerativa (abordagem *bottom-up*) (Fig. 3.12).

Num processamento *top-down*, o algoritmo inicia considerando os objetos como um único grupo que, posteriormente, é dividido em mais grupos. O algoritmo PDDP (*Principal Direction Divisive Clustering*) desenvolvido por Boley [BOL 98] é um exemplo de aplicação da decomposição de elementos em agrupamento hierárquico divisivo de coleções de documentos. O algoritmo básico constrói uma árvore binária, que é a árvore PDDP. Cada nodo da árvore é uma estrutura de dados que contém os documentos associados àquele nodo, as várias quantidades computadas a partir daquele conjunto de documentos, e ponteiros para os dois nodos filhos, caso existam.

Ao contrário, num agrupamento aglomerativo, o algoritmo considera cada objeto do grafo como um grupo de um único elemento que, posteriormente é aglomerado a outros grupos. Um exemplo de técnica aglomerativa é a do Vizinho mais Próximo, onde cada amostra única começa como um agrupamento, e a cada estágio, faz-se a fusão dos pares de agrupamentos mais próximos. Após a execução de $n - 1$ estágios, somente um agrupamento permanece, e a árvore binária resultante pode ser cortada na profundidade desejada (ou especificada) pelo usuário para levar a um particionamento específico.

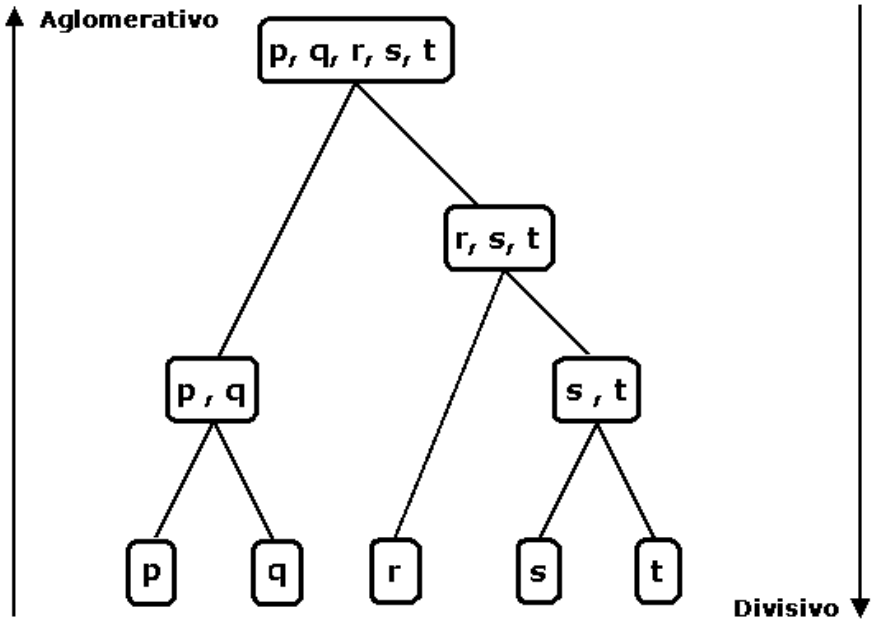


FIGURA 3.12 - Agrupamento aglomerativo e agrupamento divisivo

Um dos problemas clássicos encontrados no agrupamento hierárquico é o fato de que, na navegação de grafos agrupados, a escalabilidade é limitada, sendo que as representações bidimensionais tendem a se tornar insuficientes. Conforme aumenta a profundidade do aninhamento, aumenta o número de meta-nodos a serem abertos para navegação do grafo. Com isso, tarefas como a localização de conjuntos de dados requer grande quantidade de ações navegacionais para atingir uma localização particular num grafo.

Uma das soluções propostas é a utilização da representação tridimensional dos dados. Eades e Feng [EAD 96] apresentam uma técnica de visualização de grafos utilizando múltiplos níveis de abstração. Adotando uma representação tridimensional, cada nível é desenhado em um plano em diferentes coordenadas z, com a estrutura de agrupamento desenhada como uma árvore em três dimensões (Fig. 3.13).

Como exemplo de alguns sistemas que fazem uso de técnicas de agrupamento, Herman *et al.* [HER 2000] cita os sistemas SemNet, Narcissus e Sketch, dentre os mais antigos; e os sistemas Nicheworks, DA-TU e STARLIGHT dentre os novos sistemas desenvolvidos utilizando essas técnicas. Além desses, ainda podem ser citados sistemas como o Higres [LIS 99] e o trabalho de Marshal [MAR 2000], que discutem o uso de técnicas de agrupamento.

Além das técnicas de agrupamento hierárquico existem ainda as técnicas *flat* ou não-hierárquicas. Nestas técnicas, identificadas por Fasulo [FAS 99] como *k-clustering* ou particionamento, as informações são apresentadas num único nível. Os algoritmos de particionamento dividem os dados em vários subconjuntos. Uma abordagem de particionamento dos dados é a de pegar um ponto de vista conceitual que identifique o agrupamento com um certo modelo cujos parâmetros desconhecidos tem de ser encontrados.

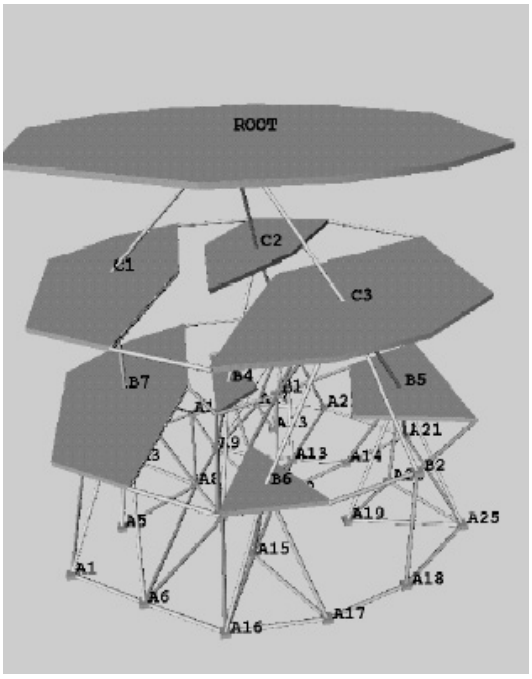


FIGURA 3.13 - Agrupamento hierárquico [EAD 96]

Métodos K-means (*K-means Methods*), são os métodos mais populares desta categoria. O método consiste da representação de *k* agrupamentos de acordo com os pontos chamados de centróide. Sendo assim, o particionamento do conjunto de dados em *k* grupos disjuntos é realizado através do agrupamento dos elementos de acordo com o nível de similaridade destes com os centros de agrupamento definidos.

Podem ser identificados ainda os algoritmos baseados em densidade, que são parecidos com as técnicas de vizinhos mais próximos citadas no agrupamento hierárquico. Nela, um agrupamento, definido como um componente denso conectado, cresce em qualquer direção que a densidade leve. Existem duas abordagens principais nesta categoria: a da densidade baseada na conectividade e a das funções de densidade.

A conectividade baseada em densidade é uma relação simétrica e todos os pontos atingíveis a partir de objetos centrais podem ser fatorizados em componentes conectados maximais servindo como agrupamentos. Os pontos que não são conectados a nenhum ponto são chamados de *outliers* por não pertencerem a nenhum agrupamento. O processamento é independente do ordenamento dos dados. Funções de densidade, por sua vez, deslocam a ênfase da computação das densidades através de pontos fixos para a computação de funções de densidade definidas sobre o espaço subjacente do atributo.

3.3.3 Layout de um grafo agrupado

Conhecendo os agrupamentos existentes dentro dos dados pode-se aplicar a redução dos elementos a serem mostrados restringindo a visualização aos próprios agrupamentos. A visualização dos agrupamentos dá ao usuário uma visão geral da estrutura do grafo, permitindo assim que se possa reter o seu contexto de maneira mais

fácil, pois implica numa menor complexidade visual do que o grafo original visualizado. Sistemas como o Higres [LIS 99] utilizam esta técnica.

Uma técnica comum de representação de agrupamentos é o uso de objetos, que são tratados como super-nodos num nível de abstração maior ou num grafo composto, no qual se pode navegar no lugar do grafo original. Geralmente estes objetos utilizados apresentam uma diferenciação visual em relação aos nodos comuns para que assim, os usuários possam facilmente notar que não se trata de um nodo original, mas um agrupamento. Marshall [MAR 2000] se utiliza destes conceitos em seu trabalho, onde os agrupamentos são representados por uma geometria diferente (Figuras 3.14 e 3.15), sendo considerados como super-nodos, podendo ser expandidos para permitir a visualização dos nodos contidos nele.

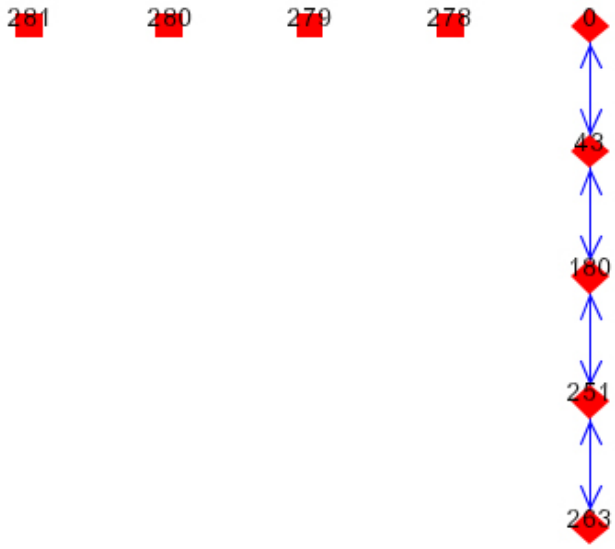


FIGURA 3.14 - Grafo com agrupamentos no sistema Royere [MAR 2000]

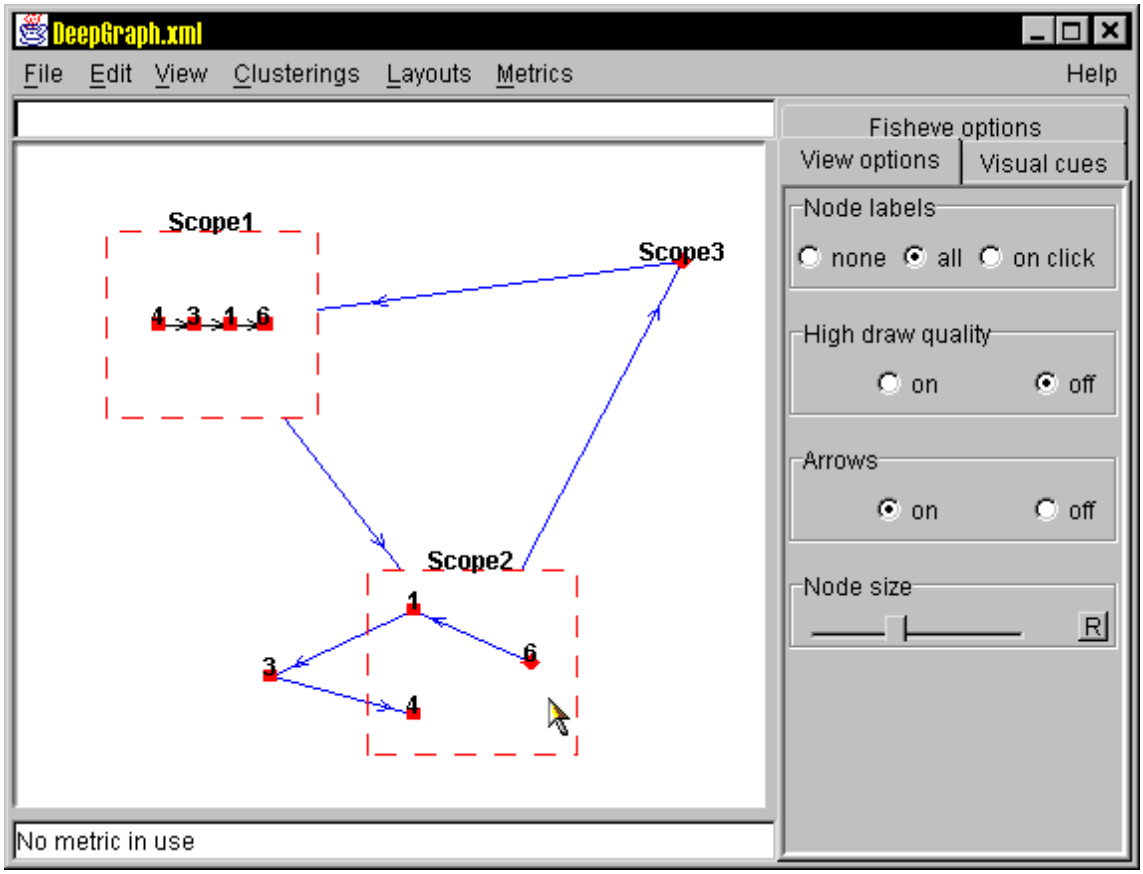


FIGURA 3.15 - Grafo com explosão dos agrupamentos no sistema Royere [MAR 2000]

3.4 Ferramentas de visualização de grafos

Diversas ferramentas foram desenvolvidas com o intuito de facilitar a tarefa de manipulação e visualização destas informações aplicando técnicas desenvolvidas com o intuito de prover uma melhor visualização das informações. Dentre elas, podemos citar os sistemas daVinci [FRO 94] e Royere [MAR 2000], além de pacotes como GraphViz [NOR 94][GAN 2000] e dos toolkits Graphlet [HIM 96], e os desenvolvidos por Dogrusoz *et al.* [DOG 2002].

O sistema daVinci é descrito por Frölich [FRO 94] como uma interface do usuário para grafos. Suporta múltiplos grafos assim como múltiplas arestas, isto é, dois nodos possuindo mais de uma aresta entre eles. A ferramenta inclui um pré-passo para a transformação dos grafos cíclicos em acíclicos através da inversão do sentido de uma das arestas em cada ciclo, permitindo assim, a visualização de grafos cíclicos.

Já o Graphlet [HIM 96], é um toolkit orientado-a-objeto para implementar editores de grafos e algoritmos de desenho de grafos. Provê uma linguagem de script para facilitar implementação de ferramentas e a customização. A linguagem script citada é o *Graph Modeling Language* (GML), que provê um mecanismo poderoso e extensível para a troca de arquivos de grafos.

O pacote Graphviz [NOR 94][GAN 2000] apresenta algumas ferramentas para a edição e visualização de grafos para as plataformas UNIX e MS-Windows. O pacote inclui as seguintes ferramentas:

- *dot*: gera layout de grafos dirigidos;
- *neato*: gera layouts de grafos não-dirigidos no modelo "*spring layout*";
- *dotty*: editor de grafos combinando o editor gráfico programável *lefty* e a ferramenta de layout de grafos *dot*;
- *tcldot*: uma interface gráfica customizável; e
- *libgraph*: a biblioteca base para as ferramentas de grafos.

O Royere [MAR 2000], por sua vez, permite a aplicação de uma variedade de técnicas de layout à estrutura visualizada, além de também apresentar uma série de métricas de agrupamento para aplicação nas estruturas. Trata-se de uma ferramenta que permite uma fácil inclusão de novas classes ou categorias de layout, ou mesmo de novas métricas para o agrupamento dos nodos. Apesar da diversidade encontrada, somente trabalha com estruturas hierárquicas e grafos acíclicos (DAG's – *Direct Acyclic Graphs*), o que limita um pouco a utilização da aplicação. Utiliza como entrada, arquivos nas linguagens GML e GraphXML.

Já Dogrusoz *et al.* [DOG 2002] desenvolveram dois toolkits para permitir que fossem facilmente integradas as capacidades de visualização de grafos a quaisquer ferramentas. Estes toolkits são:

- *Graph Layout Toolkit* (GLT) - provê interfaces para a modelagem, desenho e layout automático de grafo. O GLT provê um modelo de grafo e um framework de desenho, oferecendo ainda quatro estilos de layout: hierárquico, ortogonal, simétrico e circular;
- *Graph Editing Toolkit* (GET) - provê um nível customizável de display e edição. Além desses componentes customizáveis apresenta alguns componentes padrão e componentes específicos de aplicação, que variam entre as aplicações.

3.4.1 Linguagens de Descrição de Grafos

Até hoje foram desenvolvidas várias linguagens de descrição de grafos na tentativa de padronizar os formatos de descrição para permitir assim, intercâmbio destas informações entre as ferramentas. Entretanto, não existe uma linguagem padrão utilizada para a descrição de grafos. Dentre as iniciativas relatadas na literatura podem ser citadas as linguagens GML (*Graph Modeling Language*) e GraphXML, sendo estas, as mais difundidas entre as ferramentas atuais.

3.4.1.1 GraphXML

GraphXML é uma linguagem de descrição de grafos em XML (*Extensible Markup Language*). Os aspectos do XML tornaram possível a definição de um formato geral de intercâmbio entre o desenho de grafos e os pacotes de visualização. Este formato suporta não somente a descrição matemática pura de um grafo, como também as necessidades das aplicações de visualização de informações que utilizam estruturas de dados baseadas em grafos.

A escolha do XML foi devido a fatores como:

- flexibilidade: XML permite definir regras sintáticas claras para a especificação de uma "linguagem" específica num documento chamado *Document Type Definition* (DTD), que ainda apresenta a opção de adição de extensões;

- portabilidade: XML é utilizada por várias aplicações diferentes, sendo que dessa forma, a linguagem possui maior chance de ser aceita por outras comunidades;

- popularização de uso: já vem sendo definida uma série de especificações baseadas em XML, assim como várias ferramentas baseadas em XML estão surgindo.

Sendo assim, considerando que as necessidades da visualização de informações, vão além do puro desenho de um grafo, o GraphXML foi desenvolvido. Alguns dos aspectos do GraphXML, tal como ressaltados por Herman e Marshall [HER 2000b] são:

- inclusão de labels para nodos e arestas;

- inclusão de dados dependentes de aplicação, tanto na forma de dados armazenados diretamente como URL's;

- sentenças XML para a descrição de grafos hierárquicos, referindo tanto a arquivos locais como a descrições remotas na Web;

- descrição de grafos dinâmicos, ou seja, habilidade de descrever a evolução de grafos provendo uma ferramenta para a navegação e visualização de grafos;

- facilidade de armazenar não somente geometrias, mas atributos visuais;

- habilidade de adicionar extensões do usuário à linguagem utilizando as facilidades do XML;

Estas características tornam GraphXML uma linguagem interessante para utilização na descrição de grafos, uma vez que oferece uma série de possibilidades, além da flexibilidade para adequação às necessidades de visualização decorrentes.

3.4.1.2 GML

GML, ou *Graph Modeling Language*, é o resultado de uma iniciativa visando o desenvolvimento de um formato de arquivos único para a descrição de grafos. Foi utilizada pela primeira vez dentro do sistema Graphlet [HIM 96] e desenvolvida tendo como principais requerimentos:

- independência de plataforma – obtida através do formato de representação ASCII, que além de dar portabilidade também provê maior simplicidade;
- facilidade de implementação – obtida através da estrutura simples do GML, que consiste de pares de valores-chave hierarquicamente organizados;
- capacidade de representação de estruturas de dados arbitrárias, uma vez que os programas têm a necessidade de anexar os dados específicos de nodos e arestas;
- flexibilidade, no sentido de não haver a necessidade de uma ordem específica de declarações.

Uma das dificuldades advindas das características descritas anteriormente se refere à questão da possibilidade de representação de estruturas arbitrárias. Ou seja, GML é extensível de maneira que podem ser agregadas definições de características das estruturas de dados que não estejam ainda definidas no formato padrão de arquivos. Como resultado desta extensão, um dos objetivos da linguagem, que era a portabilidade dos arquivos entre as ferramentas pode ser perdido, uma vez que descrições geradas numa ferramenta não serão lidas por outra aplicação.

4 Estudo de caso: o sistema Hyper-Automaton

Este capítulo destina-se à descrição do sistema Hyper-Automaton [MAC 2000], que foi utilizado como estudo de caso para esta dissertação. O sistema é o resultado de um trabalho desenvolvido dentro do Projeto e-Automaton, sendo tema de uma dissertação de Mestrado do PPGC da Universidade Federal do Rio Grande do Sul.

4.1 Descrição do sistema

O Hyper-Automaton é um sistema que tem como objetivo principal dar suporte à publicação de materiais instrucionais (especificamente, cursos e tutoriais, exercícios e provas adaptativas) na Web. Baseado numa arquitetura cliente-servidor, o Hyper-Automaton é um sistema semi-automatizado que, apesar de ter sido desenvolvido no contexto de educação à distância, pode ser aplicado a sistemas de hipertexto como um todo.

O trabalho foi desenvolvido enfocando um mecanismo básico para a estruturação de um sistema hipertexto através de um grafo dirigido com informações associadas aos nodos e arcos representando as ligações. Nesta representação, um hiperdocumento é resultado da composição de vários tipos de arquivos, trechos de textos em HTML, figuras, simuladores em Java e outros recursos hipermídia interligados por meio de hiperlinks de acordo com a estrutura definida no autômato. Considerando que cada página WWW corresponde a um estado (ou nodo) e cada *link* corresponde a um arco etiquetado (ou transição entre os estados), sendo que este arco é definido por uma função que relaciona os estados, pode-se dizer que a estrutura de um hiperdocumento corresponde a um autômato finito.

Assumindo cada estado do autômato como sendo correspondente a cada página do hiperdocumento, tem-se um autômato finito com saída (Máquinas de Mealy e Moore). Neste caso, além das definições apresentadas anteriormente, os fragmentos de informação relacionados a cada estado (ou nodo) passam a ser considerados como as saídas de cada um deles. A utilização deste tipo de representação foi proposta com o intuito de criar uma estrutura funcional para a apresentação de materiais na Web, visando assim facilitar a implementação, a manutenção e o reuso de materiais já criados.

Cada um destes arquivos e páginas relacionados faz parte do conjunto de símbolos de um alfabeto de saída que, por sua vez, são utilizados pela função de saída relacionada a cada estado ou transição. A função de saída é a responsável pela montagem dos hiperdocumentos que serão apresentados aos usuários. Todos os fragmentos de informação utilizados (ou seja, arquivos e páginas selecionados para exibição) definidos no alfabeto de saída são armazenados numa hiperbase independente da estrutura de controle hipermídia.

Já os links entre cada uma das páginas passam a não ser mais estáticos e codificados no documento HTML, e sim, implementados como funções de transição dos autômatos. Trabalhando desta forma, o sistema provê liberdade ao usuário-autor dos hiperdocumentos de alterar a estrutura dos autômatos sem modificar as páginas HTML e vice-versa. Outra vantagem decorrente é a possibilidade de reutilização das unidades

de informação já definidas, sem que haja interferência de uma estrutura de hiperdocumento sobre outra. Isso ocorre justamente devido à implementação das funções de transição, o que provê a independência dos *links* dos hiperdocumentos relacionados em cada autômato.

Sendo assim, pode-se dizer que o modelo de representação de hipertextos foi estendido, de forma que:

- um nodo pode representar um conjunto de páginas Web, e não apenas uma;
- uma função agrupa páginas em unidades que podem ser exibidas;
- esta função pode ser associada a um nodo ou transição.

Utilizando estes conceitos, um sistema com interface visual em HTML foi implementado, provendo ao usuário uma interpretação tangível dos conceitos de Autômatos Finitos (Máquinas de Mealy e Moore) aplicados. Os elementos do alfabeto de saída são anotados como unidades de informação enquanto que o resultado da função de próxima saída é o hipertexto visualizado na janela do browser. Os alfabetos de entrada que etiquetam as transações no autômato são mostrados como links HTML que um usuário pode selecionar. O próximo link é uma projeção da função de próximo estado no ambiente hipertexto.

O sistema Hyper-Automaton tem arquitetura cliente-servidor, sendo que o servidor é formado por uma série de programas CGI alocados num servidor HTTP e oferece os serviços de controle das estruturas dos autômatos, assim como o acesso à hiperbase com os fragmentos de informação que poderão ser utilizados. Os clientes do sistema podem ser qualquer tipo de navegador da WWW que tenha suporte a applets Java. A Figura 4.1 apresenta um desenho esquemático da arquitetura do sistema.

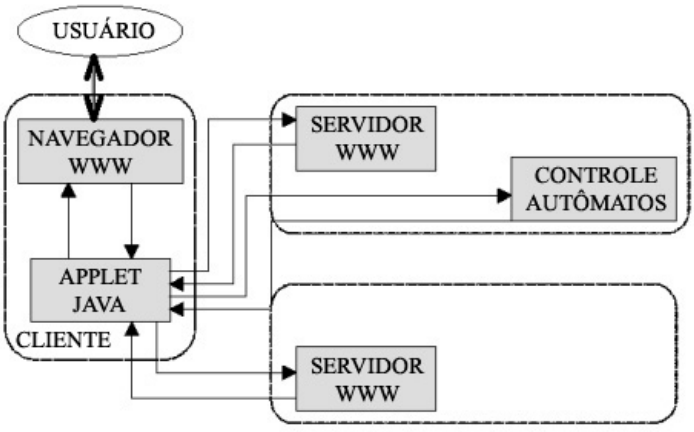


FIGURA 4.1 - Esquema da arquitetura do sistema [MAC 2000]

O sistema apresenta uma série de vantagens, listadas a seguir. Observa-se que algumas delas são vantagens levando em conta um ambiente de educação à distância:

- possibilidade de utilização de conteúdo multimídia;
- ambiente amigável e de fácil manipulação;

- ambiente integrado, pois é possível visualizar diferentes mídias dentro do próprio navegador (*browser*)

- permite a integração com outros serviços da *Internet*. Com a utilização da interface comum disponibilizada pelos navegadores é possível acessar serviços como o *ftp*, *telnet*, *newsgroup* e correio eletrônico;

- uso de *hyperlinks*, para melhor estruturação do conteúdo;

- favorece uma educação ativa, ou seja, o aluno atua no processo de descoberta dos conhecimentos;

- possibilidade de acesso às informações através de diferentes plataformas de *hardware* e *software*;

- flexibilidade de horário, o material disponibilizado pode ser estudado em qualquer horário.

4.2 Análise da ferramenta

Dadas as especificações da ferramenta e as vantagens oferecidas pela mesma, deve-se agora apresentar uma análise mais aprofundada da mesma destacando falhas existentes ou necessidades que possam ser supridas através da aplicação de técnicas inerentes de visualização.

A primeira análise diz respeito à interface com o usuário do Hyper-Automaton. O Hyper-Automaton possui uma interface HTML simples, semi-automatizada e de fácil manuseio (Fig. 4.2). Como se pode ver, os autômatos são definidos de uma maneira simplificada utilizando tabelas para relacionar cada estado às saídas a serem geradas (unidades de informação), assim como a definição dos destinos das transições que serão inclusas em cada página. No entanto, considerando conceitos de visualização de informações esta suposta “simplicidade” pode ser também o ponto causador de uma série de problemas na utilização da ferramenta.

Estado	Saída	Página Anterior	Próxima Página	Avançado	Capítulo Anterior	Próximo Capítulo
0	RG;C1Fig;C1		1			43
1	RG;C1T1B	0	2			43
2	RG;C1T1C	1	3			43
3	RG;C1T1D	2	4			43
4	RG;C1T2A	3	5			43
5	RG;C1T2B;C	3	6			43
6	RG;C1D2B;C	5	7			43
7	RG;C1O2D	6	8			43
8	RG;C1T2C;C	7	9			43
9	RG;C1D2D;C	8	10			43
10	RG;C1D2E;C	9	11			43
11	RG;C1D2F;C	10	12			43
12	RG;C1D2G;C	11	14			43
13	RG;C1D2H;C	12	14			43
14	RG;C1T2D;C	12	15			43
15	RG;C1D2K;C	14	16			43

FIGURA 4.2 - Forma de edição de um autômato no Hyper-Automaton

O método apresentado acima, além de custoso, no que diz respeito ao tempo de definição das estruturas, dá margem à ocorrência de uma série de erros. Não há como negar que os usuários-autores dos hiperdocumentos vão dispender muito mais tempo do que deveriam numa tarefa considerada relativamente simples, uma vez que todos os fragmentos de informação a serem utilizados nos hiperdocumentos já foram criados. Conforme o aumento da complexidade das estruturas de informações, maior será o tempo dispendido nas atividades de definição dos autômatos finitos com saída.

Com relação à margem de erros, vamos utilizar como exemplo, alguns tipos de documentos visados na proposta de desenvolvimento do sistema, livros e tutoriais. Estruturas como estas envolvem a utilização de uma grande quantidade de informação. Ao trabalhar com a forma de edição apresentada atualmente, fica claro que em algum momento ficará impossível realizar um gerenciamento efetivo das informações. Com a quantidade de linhas e colunas cada vez maior, fica muito mais fácil cometer um engano, porém, muito mais difícil identificá-lo. Além disso, a confusão mental causada pela quantidade de informação, somente contribuirá mais para o desgaste do usuário, pois a estruturação do hiperdocumento mentalizada pelo autor deve ser traduzida para a descrição na tabela de estados.

Já tarefas como as de gerenciamento e verificação das estruturas de hiperdocumentos geradas também tendem a se tornar mais complexas conforme o aumento do tamanho das estruturas. Isso porque tarefas como a verificação das informações relacionadas implicaria na visualização de todo o hiperdocumento, necessitando da verificação link a link, ou mesmo da observação minuciosa de cada item da tabela. Neste caso, deve-se dar atenção ao fato de que os usuários ainda deverão identificar os rótulos utilizados pelas saídas, para que possam recordar quais fragmentos

são relacionados a cada nodo. Num livro, que se trata de uma estrutura complexa, o tempo a ser empregado na atividade de verificação seria enorme, ou seja, os objetivos iniciais da ferramenta que eram de dinamizar o processo de criação e gerenciamento de hiperdocumentos não seriam atingidos; na realidade, estariam perdidos.

Atividades como as citadas acima seriam muito mais efetivas se efetuadas de forma gráfica, mais especificamente, se fizessem uso dos recursos da edição e visualização de grafos. Através da seleção ou desenvolvimento de técnicas adequadas, o processo de verificação e localização das informações em estruturas complexas se torna muito mais simples.

Assim, com base na literatura, pode-se investigar a técnica de visualização mais apropriada para a visualização de autômatos finitos com saída. Herman *et al.* [HER 2000] apresenta um resumo dos tipos de visualização mais conhecidos e utilizados atualmente e Battista *et al.* [BAT 99] reúnem uma variedade de algoritmos de desenho de diagramas, que incluem layouts diversos e representações visuais variadas para nodos e arestas.

Já para o processo de edição, podem ser utilizadas ferramentas como os sistemas daVinci [FRÖ 94], Graphlet [HIM 96] [HIM 2000] e Royere [MAR 2000], que permitem a edição de grafos de um modo geral. Os toolkits GLT/GET [DOG 2002] e GraphViz [NOR 94] [GRA 2000] oferecem ferramentas para o desenvolvimento de interfaces de edição e visualização de grafos.

O próximo capítulo detalha a proposta de visualização de grafos para o Hyper-Automaton.

5 Aplicação de Técnicas de Visualização de Grafos ao Hyper-Automaton

Considerando as características do Hyper-Automaton e suas necessidades, optou-se por desenvolver inicialmente uma técnica para a visualização de autômatos finitos com saída, utilizando as funcionalidades de algumas ferramentas já existentes. Desta forma, tarefas como as de verificação e visualização das estruturas geradas já seriam beneficiadas.

Em um segundo momento, uma ferramenta que permita a edição dos autômatos finitos com saída deve ser implementada, possibilitando a geração automática dos arquivos descritores do Hyper-Automaton, substituindo a forma de edição utilizada atualmente. A nova ferramenta, além da funcionalidade de edição, também deverá agregar a técnica de visualização aqui proposta para os fins de gerenciamento e verificação já citados.

A seguir, detalhamos a investigação de técnicas de visualização para a estrutura de representação de hiperdocumentos utilizada pelo Hyper-Automaton.

5.1 Estrutura das informações

Para que seja possível gerar a visualização dos autômatos finitos com saída, é necessário que, antes de tudo, seja possível obter as informações das estruturas de informação utilizadas pelo Hyper-Automaton. As informações utilizadas pelo sistema são todas armazenadas em arquivos descritores, sendo que foram identificados três tipos de arquivos que serão necessários para o desenvolvimento da visualização. Todos os arquivos são do formato (*.fl) com dados ASCII, portanto, arquivos simples e de fácil manipulação:

- arquivo de saídas: identificados como *saidas.fl*, estes arquivos são responsáveis por relacionar todos os fragmentos de informação contidos na hiperbase com os seus respectivos identificadores, que serão utilizados na ferramenta;

- arquivos de alfabeto de entrada: este arquivo será responsável pelo armazenamento do alfabeto de entrada do autômato finito que será visualizado. Informações como rótulos das transições que podem estar disponíveis para os hiperdocumentos são identificadas neste alfabeto de entrada. Vale ressaltar que cada autômato finito gerado tem seu próprio alfabeto de entrada;

- arquivos descritores: São os arquivos que contém a descrição dos estados do autômato, as saídas relacionadas a cada um deles, além dos relacionamentos (ou transições) referentes a cada estado.

Sendo assim, a tarefa inicial corresponderá à leitura e armazenamento de todas as informações fornecidas nestes arquivos para posteriormente, realizar a tradução destas informações para o formato específico da ferramenta de visualização de grafos utilizada.

Desta forma, foi criada uma estrutura intermediária para o armazenamento destas informações. Esta estrutura permite tanto o armazenamento de informações de grafos como de autômatos, uma vez que ambos possuem estruturas semelhantes, sendo os objetos principais, os nodos (ou estados) e as arestas (ou transições) e as saídas os elementos específicos das estruturas de autômatos. A Figura 5.1 apresenta a estrutura de informações adotada na forma das classes utilizadas na implementação. A classe *Output* foi inclusa na estrutura de armazenamento de maneira a permitir o relacionamento de saídas tanto aos estados quanto às transições de um autômato, quando necessário.Nada impede que estruturas de grafos triviais sejam armazenadas nesta estrutura de informação.

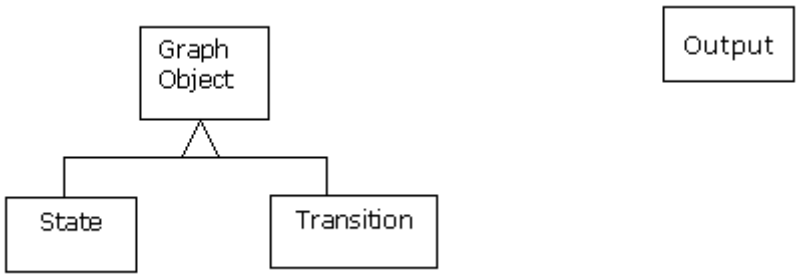


FIGURA 5.1 - Estrutura de armazenamento das informações

5.2 Tradução dos dados

Antes da integração das técnicas de visualização ao sistema Hyper-Automaton foram geradas visualizações em algumas ferramentas existentes, para avaliação das propostas.

Para efeito de realização de testes iniciais, foi utilizada a ferramenta Dotty do pacote GraphViz [NOR 94][GAN 2000]. Esta ferramenta foi selecionada devido à facilidade de sua estrutura de descrição, dando maior rapidez à esta etapa de desenvolvimento. A ferramenta Dotty utiliza como entrada um arquivo numa linguagem específica armazenada no formato **.dot*, que foi desenvolvida especificamente para a ferramenta. Esta linguagem possui três tipos de itens: grafos nodos e arestas. O grafo principal, ou mais externo, pode ser tanto um grafo dirigido, como não dirigido. Dentro de um grafo principal, podem ser definidos subconjuntos de nodos e arestas.

Numa segunda etapa, após a análise das necessidades de visualização e definição da técnica a ser proposta, foi utilizada a ferramenta Royere [MAR 2000]. Esta ferramenta foi selecionada justamente devido à sua estrutura, que permite a extensão da ferramenta. Além disso, a ferramenta já provê uma série de algoritmos de layout já implementados, além de várias outras funcionalidades não implementadas no Dotty. A vantagem do Dotty com relação ao Royere é no que diz respeito à visualização de estruturas de grafos cíclicos. O Royere possui uma classe de verificação que impede que estruturas cíclicas sejam visualizadas para que possa aplicar os algoritmos de layout que têm à disposição. Já o Dotty realiza este tratamento das estruturas por utilizar apenas um algoritmo de layout.

O Royere aceita os formatos GML e GraphXML para a descrição das informações, possuindo uma série de algoritmos de layout de grafos. Dentre as opções apresentadas foi selecionado o GraphXML, justamente pela facilidade de extensão (apesar de o GML também possibilitá-la) e também das potencialidades do próprio XML. Aliado a está justificativa, pesou também o fato da própria equipe do projeto e-Automaton estar estudando a utilização do XML numa futura versão da ferramenta.

Como a categoria de estruturas tratada aqui é a de autômatos finitos com saída, foi necessária ainda a realização de uma extensão da DTD (*Document Type Definition*) do GraphXML para que este pudesse aceitar as descrições das saídas. Primeiro, na especificação de atributos do elemento *node* (Figura 5.2), foi inserido o atributo *hasOutput*, do tipo booleano, para que a ferramenta realize a tarefa de extração destas informações somente quando esta propriedade for setada como verdadeira.

```
<!ELEMENT node (%common-elements;
                %nodeExtensions;
               |style|position|size|transform|subgraph-style)*>
<!ATTLIST node
   xmlns:xlink          CDATA                #FIXED
"http://www.w3.org/1999/xlink/namespace/"
   name                 CDATA                #REQUIRED
   isMetanode           (true|false)         "false"
   hasOutput            (true|false)         "false"
   xlink:type           CDATA                #FIXED      "simple"
   xlink:role           CDATA                #FIXED      "Reference to graph"
   xlink:title          CDATA                #IMPLIED
   xlink:show           (new|parsed|replace) #FIXED      "new"
   xlink:actuate        (user|auto)          #FIXED      "user"
   xlink:href           CDATA                #IMPLIED
   class                CDATA                #IMPLIED
>
```

FIGURA 5.2 - Atributos do elemento node

Em seguida, também foi incluso o elemento *output*, com o intuito de armazenar todas as informações relacionadas a esta categoria de objeto não prevista dentro do GraphXML. Poderia ter sido utilizada a descrição das saídas como nodos, porém, para fins de maior clareza da definição, optou-se pela extensão do modelo proposto. Trata-se do elemento mais simples dentre todos os definidos anteriomente, uma vez que a maioria dos atributos que poderia ser aceita já fora definida em *%common-elements*. Estes atributos correspondem a informações como rótulo, dados, referência dos dados e propriedades. A Figura 5.3 apresenta a definição do elemento *output*.

```
<!ELEMENT output (%common-elements;
                  %outputExtensions;
                  |style|size|position)*>

<!ATTLIST exit
  name  CDATA   #IMPLIED
  class  CDATA   #IMPLIED
>
```

FIGURA 5.3 - Definição do elemento output

Dadas as extensões da DTD GraphXML descritas em decorrência da seleção da ferramenta Royere apresenta-se portanto o modelo de funcionamento da visualização

esquematizado na Figura 5.4. O processo de visualização se dará portanto através das seguintes etapas:

Geração dos arquivos descritores da estrutura na ferramenta Hyper-Automaton;

- Armazenamento das informações necessárias contidas nos arquivos descritores de saídas, alfabeto de entrada e do programa;
- Agrupamento das informações de acordo com métrica estabelecida;
- Geração dos arquivos GraphXML (e arquivos .dot para o caso de utilização da ferramenta Dotty) com descrição do autômato agrupado e dos agrupamentos e saídas relacionadas a cada estado do autômato finito;
- Visualização da estrutura na ferramenta Royere.

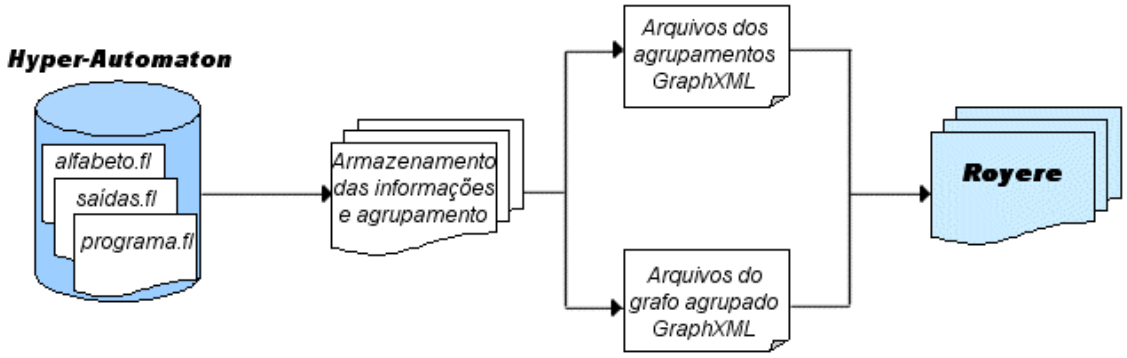


FIGURA 5.4 - Modelo de funcionamento para a visualização dos autômatos finitos

5.3 Visualização de autômatos finitos com saída

A visualização dos autômatos tem como objetivo suprir a carência de uma representação gráfica para auxiliar principalmente as tarefas de verificação e visualização para o gerenciamento das informações do Hyper-Automaton. Espera-se obter como impacto da utilização destes recursos, reduzir a tendência à ocorrência de erros, dinamizar os processos relacionados, além de imprimir maior velocidade às tarefas.

Visando realizar uma análise inicial dos hiperdocumentos gerados pelo Hyper-Automaton, foram realizadas visualizações dos diferentes tipos de autômatos gerados pelo programa. Ou seja, foram analisadas as visualizações de livros ou tutoriais, de provas online, assim como as estruturas de exercícios adaptativos.

No caso de livros e tutoriais, foi selecionado um tutorial[1] específico, composto por 281 unidades de informação (ou seja, estados), previamente criadas utilizando o Hyper-Automaton e com suas informações devidamente armazenadas em um arquivo

[1] Tutorial disponível em http://teia.inf.ufrgs.br/

descritor. Finalizada a leitura, os dados armazenados foram traduzidos para o formado desejado (no caso um arquivo **.dot*) para visualização no Dotty. Vale ressaltar que nesta primeira etapa foram visualizados apenas os estados da estrutura, a fim de avaliar o nível de complexidade obtido na visualização. A Figura 5.4 apresenta a primeira visualização da estrutura do autômato finito citado.

Logo em seguida, também foram geradas visualizações dos demais tipos de estruturas que podem ser gerados pela ferramenta; ou seja, provas online e exercícios adptativos, que podem ser observados na Figura 5.5 e na Figura 5.6.

Caso contrário, ou seja, caso deseje observar as informações com maior nível de detalhe, terá de fazer o uso intensivo de recursos como rolagem (Fig. 5.7) para percorrer a visualização e localizar de que forma os nodos relacionados estão organizados. Além de ser uma tarefa demorada, o uso destas funcionalidades implica num risco contínuo de perda do contexto de navegação, já que a quantidade de relacionamentos e o próprio layout imposto pela ferramenta não colaboram para a execução da tarefa.

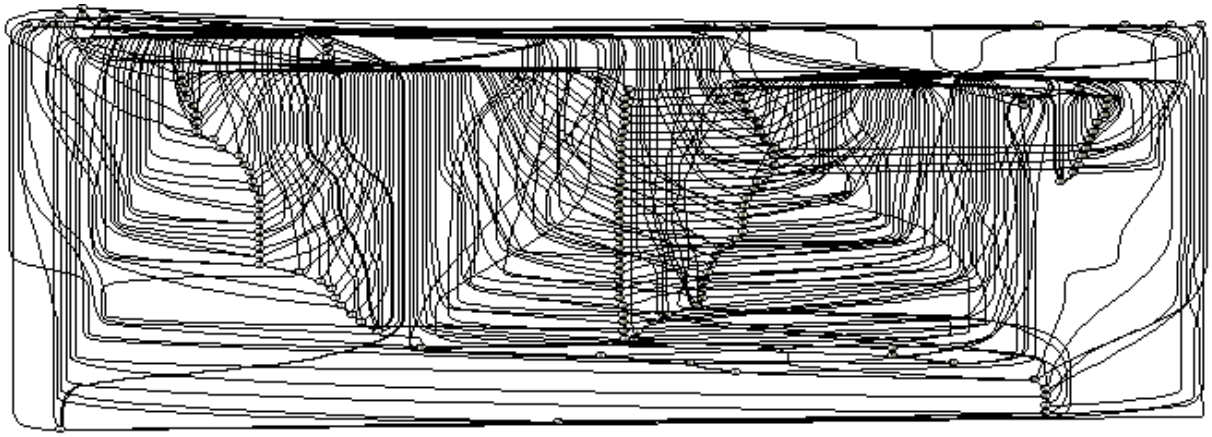


FIGURA 5.5 - Visualização do autômato de um tutorial de 281 estados

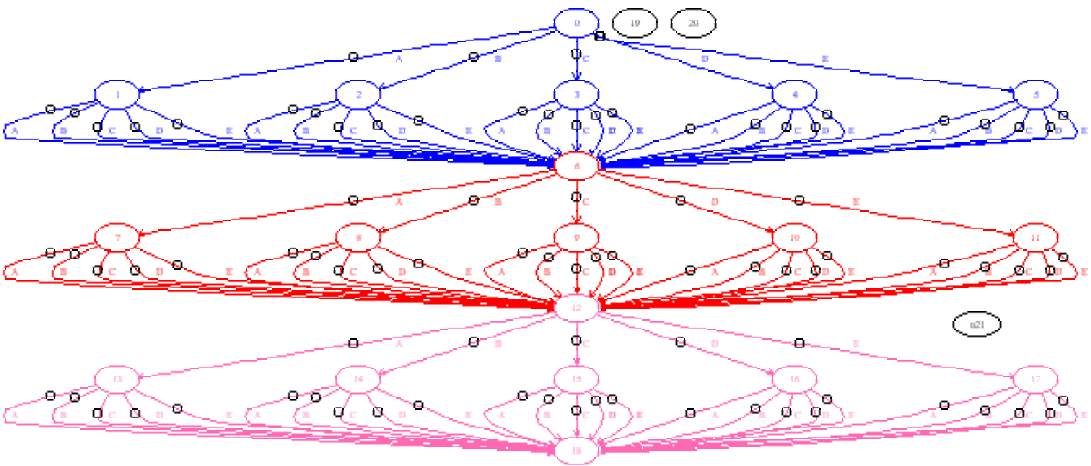


FIGURA 5.6 - Estrutura de prova adaptativa

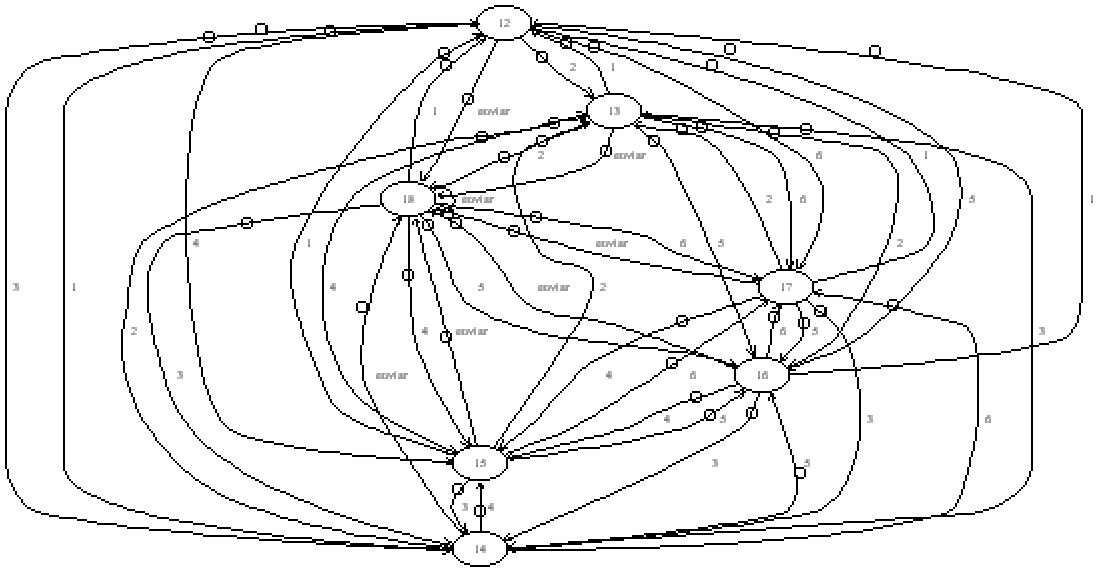


FIGURA 5.7 - Exercício adaptativo online

Tendo gerado todas as visualizações, pode-se dizer que resultado obtido na categoria de livros e tutoriais foi um autômato de grande complexidade visual, mesmo não estando representadas as saídas de cada estado. A quantidade de relacionamentos existentes provocou uma visualização impraticável se forem considerados os objetivos inicialmente propostos. Utilizando a visualização da maneira como foi apresentada, caso o usuário deseje ter uma visão geral da estrutura, terá o desenho apresentado na Figura 5.4, e provavelmente não cumprirá a tarefa de transmitir o contexto da informação.

Caso contrário, ou seja, caso deseje observar as informações com maior nível de detalhe, terá de fazer o uso intensivo de recursos como rolagem (Fig. 5.7) para percorrer a visualização e localizar de que forma os nodos relacionados estão organizados. Além de ser uma tarefa demorada, o uso destas funcionalidades implica num risco contínuo de perda do contexto de navegação, já que a quantidade de relacionamentos e o próprio layout imposto pela ferramenta não colaboram para a execução da tarefa.

- Considerando que os autômatos que representam livros e tutoriais tendem a possuir tamanhos até maiores, deve-se desenvolver técnicas que:

- reduzam a complexidade visual da informação;

- permitam a visualização das saídas relacionadas aos autômatos, para facilitar a compreensão do conteúdo da estrutura de informação.

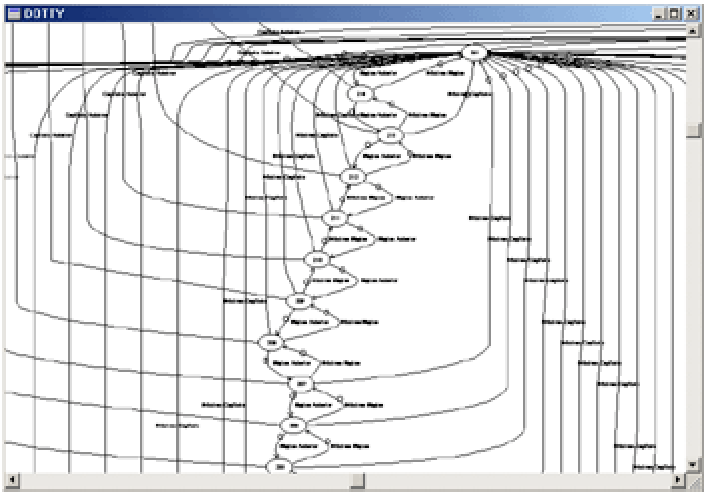


FIGURA 5.8 - Visualização mais detalhada no Dotty

Com relação às estruturas de provas online e exercícios adaptativos o panorama se mostrou um pouco mais otimista. As provas online já apresentaram um layout inicial considerado legível e aceitável. Podem ser desenvolvidas metodologias complementares para a visualização dos dados relacionados, ou mesmo alguma técnica que favoreça a compreensão das informações nela contidas.

A estrutura dos exercícios adaptativos, pelo que foi observado, consiste de um grafo completo, faltando a solução da visualização das saídas. Para prover um ordenamento visual, podem ser sugeridos layouts como os de anel, para a visualização destas informações. Possivelmente estes problemas serão solucionados facilmente.

Dados os resultados e análises das técnicas apresentadas, optou-se pelo desenvolvimento de técnicas de visualização para as estruturas de livros e tutoriais. Sendo assim, os objetivos principais a serem atingidos pelas técnicas são:

- redução da complexidade visual;
- compreensão e visualização da estrutura de navegação da informação;
- visualização das informações das saídas dos autômatos;
- evitar a perda do contexto de navegação por parte dos usuários;
- compreensão da estrutura da informação.

5.3.1 Seleção da técnica

A compreensão da estrutura de navegação das informações pode ser facilitada de diferentes maneiras: através do realce dos elementos da estrutura ou através da redução da complexidade visual. O realce dos elementos pode ser efetuado através de técnicas como a de coloração. Técnicas de coloração consistem da atribuição de cores a nodos e

arestas de um grafo. A redução da complexidade visual, por sua vez, pode ser obtida através de aplicação de técnicas de agrupamento, filtragem ou poda.

Com o objetivo de análise, a técnica de coloração foi aplicada para verificação da eficiência de sua utilização. A coloração foi aplicada de forma a identificar os diferentes capítulos existentes dentro de um livro ou tutorial. A Figura 5.8 é uma demonstração do resultado obtido através da aplicação desta técnica na ferramenta Dotty. Neste exemplo, foram alterados alguns dos parâmetros de visualização dos nodos para que, ao final, obtivéssemos a visão dos capítulos contidos no tutorial. Não houve métrica específica aplicada neste caso, já que a experiência foi realizada com o objetivo de verificar a eficácia do resultado final para a visualização.

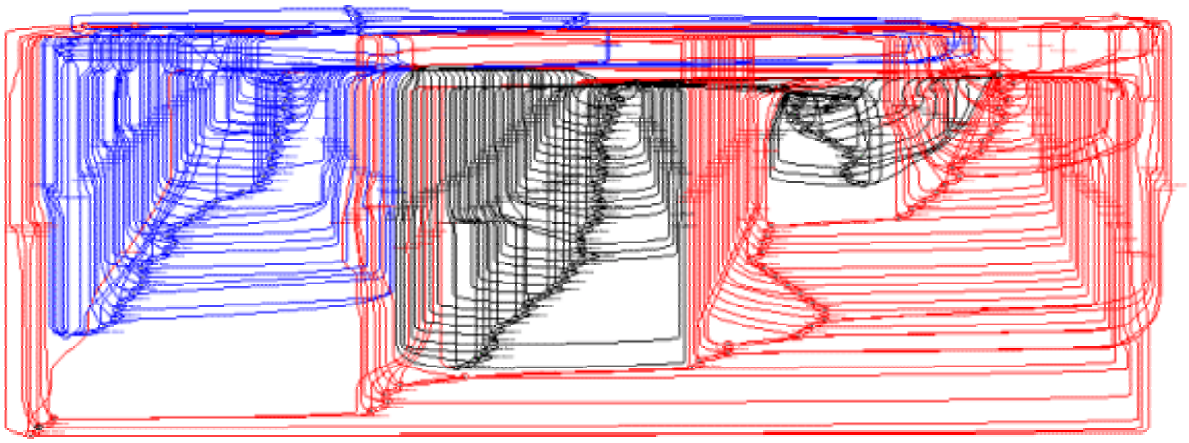


FIGURA 5.9 - Aplicação da técnica de coloração à estrutura de um tutorial

Apesar de ser um recurso eficiente para a identificação das macro-estruturas contidas no autômato, a coloração não se mostrou suficiente. A técnica empregada de fato possibilita a identificação visual dos capítulos do tutorial, porém não permite a identificação nem a compreensão das estruturas de navegação, uma vez que a mesma mantém o mesmo nível de complexidade apresentado anteriormente. Isto se deve, em grande parte, à grande quantidade de arestas existentes nos grafos. Ou seja, caso o aspecto da complexidade visual não seja devidamente tratado, haverá poucos benefícios com a utilização da coloração.

No caso da aplicação da segunda alternativa, a redução da complexidade visual, o processo a ser aplicado é baseado na diminuição do número de nodos e conexões exibidos na visualização. Esta redução pode se dar através da aplicação de técnicas como a filtragem, a poda e o agrupamento.

Uma operação de filtragem elimina os nodos que não satisfazem um determinado critério, sendo que geralmente é realizada tendo como base o conteúdo dos nodos. A poda, por sua vez, é realizada através da eliminação dos elementos que estejam longe da área de interesse do usuário. Caso a técnica fosse aplicada a um autômato, por exemplo, os nodos cujo caminho a partir do nodo atual fossem mais longos que um determinado limiar poderiam ser eliminados. O agrupamento, por outro

lado, mantém a informação original, exibindo-a com um nível de abstração mais elevado. Ou seja, a estrutura da informação é preservada, havendo apenas a diminuição da complexidade visual devido aos agrupamentos formados.

Sendo assim, a aplicação de uma técnica de agrupamento se mostrou mais apropriada, já que não elimina qualquer elemento da visualização, mas cria uma abstração das mesmas. Através da utilização de métricas adequadas, é possível apresentar aos usuários a estrutura de capítulos implícita na estrutura, aumentando a capacidade de compreensão dos usuários da aplicação, além de reduzir a complexidade visual apresentada até o momento.

A próxima seção apresenta uma proposta de agrupamento a ser aplicada, considerando que inicialmente somente será observada a estrutura geral do grafo. Ou seja, inicialmente é desconsiderada a existência das saídas relacionadas a cada um dos estados. Num segundo momento, após o desenvolvimento de técnicas de redução da complexidade visual será trabalhado o desenvolvimento de uma técnica para a visualização das saídas do autômato.

5.4 Técnicas propostas

5.4.1 Redução da complexidade visual através de agrupamento

O agrupamento é um processo que se baseia na limitação do número de elementos que podem ser visualizados. A menor complexidade visual faz com que o usuário compreenda e assimile as informações mais facilmente. A técnica é bastante utilizada para fornecer mapas de *overview* de estruturas, caso que se mostra bastante apropriado, uma vez que estão sendo consideradas estruturas de livros e tutoriais. Além disso, com a redução da quantidade de informação visualizada, ainda haverá melhora da performance do sistema.

Conforme já discutido, uma operação de agrupamento pode ser realizada com base na estrutura do grafo ou no conteúdo da informação ali representada. Sabe-se que agrupamentos baseados em conteúdo possuem resultados até melhores que os baseados em estrutura, porém, pelo fato de não permitirem a generalização da técnica, optou-se pela segunda alternativa. A base da técnica de agrupamento estrutural corresponde à localização de estruturas implícitas no autômato. Ou seja, deve-se encontrar um critério de agrupamento que possibilite a localização dos sub-grafos correspondentes às estruturas contidas no autômato finito.

Sabendo que a estrutura gerada se trata de um multigrafo desconectado, pois existem elementos isolados, pode-se resumir todo o processo através das seguintes etapas:

1. Extração do sub-grafo conectado S contido em G. Os elementos isolados extraídos nesta etapa somente serão apresentados na visualização final. Eles serão desconsiderados nesta etapa com o objetivo de facilitar a análise da estrutura restante.
2. Localização do conjunto A de pontos de articulação e do conjunto B de blocos do sub-grafo S (veja Figura 5.9).

3. Determinação dos blocos candidatos ao agrupamento a um ponto de articulação.
4. Agrupamento dos pontos de articulação contidos em A aos blocos de B de acordo com métrica aplicada.
5. Visualização do grafo com os elementos agrupados e os nodos isolados presentes.

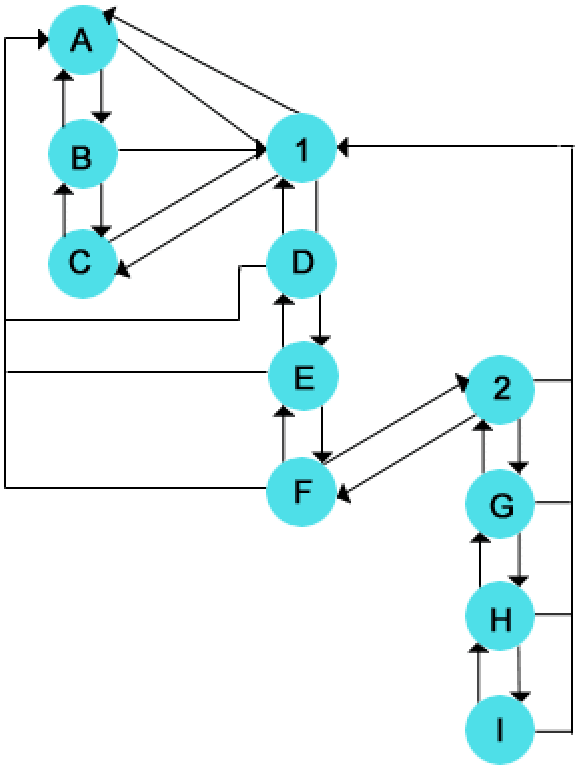


FIGURA 5.10 - Exemplo de um sub-grafo de um tutorial

Um sub-grafo S de um grafo G = (V, E) pode ser definido como S = (V', E'), de modo que V' esteja contido em V, E' esteja contido em E, e os terminais de qualquer aresta em E' também estejam em V'. Sendo assim, S é um sub-grafo conectado se houver pelo menos uma conexão entre os componentes *v* de V. A tarefa inicial envolvida consiste, portanto, na localização destes sub-grafos. Os componentes (ou elementos) desconectados de V inicialmente são ignorados. Consideremos como exemplo de subgrafo, o grafo apresentado na Figura 5.9.

A segunda etapa corresponde à localização do conjunto A = $\{v_1,.. , v_n\}$ de elementos considerados como articulações do grafo S extraído de G e do conjunto B = $\{b_1, b_2, ..., b_n\}$ de blocos do sub-grafo S onde, $b_n = (V_b, E_b)$ e $V_b \subset V'$ e $E_b \subset E'$. Na visualização de estruturas de livros e tutoriais, os blocos identificados correspondem aos capítulos especificados nos autômatos. Os pontos de articulação, por sua vez, correspondem ao ponto inicial, ou nodo de início das sub-estruturas encontradas. Na estrutura da Figura 5.9, temos que o conjunto de pontos de articulação A = {1, 2} e o conjunto de blocos B = {BLOCO 1, BLOCO 2, BLOCO 3}, onde BLOCO 1 = {A, B, C}, BLOCO 2 = {D, E, F} e BLOCO 3 = {G, H, I}, como pode ser visto na Figura 5.10.

Tendo identificado estas informações, a tarefa atual consiste na especificação de uma métrica para o agrupamento destes pontos de articulação aos seus respectivos capítulos. Espera-se desta forma, que a estrutura organizacional do autômato (ou seja, a estrutura de capítulos implícita) em questão fique mais clara aos usuários do sistema.

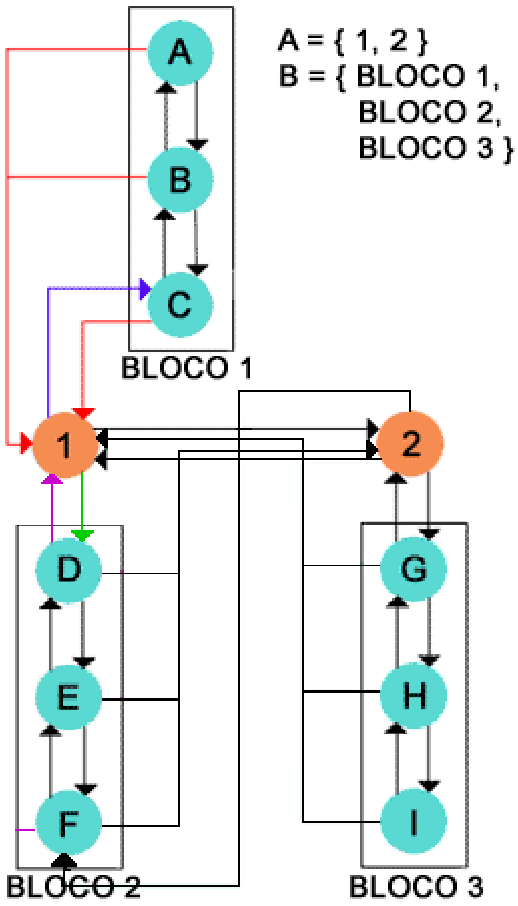


FIGURA 5.11 - Blocos e articulações

Seja $Adj(b_n) = Adj(v_1) \cap Adj(v_2) \cap ... \cap Adj(v_n)$, onde $\{v_1, v_2, ..., v_n\} \in b_n$ e Adj(v) a lista de adjacência do ponto de articulação em questão, tem-se que um bloco b_n adjacente a *v* é canditato ao agrupamento a um ponto de articulação *v* se ambos possuírem nodos adjacentes em comum, ou seja, se $(Adj(b_n) - v) \subset Adj(v)$. O agrupamento é definido, então, a partir da métrica aplicada a cada um destes blocos. Sendo assim, na Figura 5.10, temos que Adj(1) = {C, D, 2}, Adj(2) = {F, G, 1}, Adj(BLOCO 1) = {1}, Adj(BLOCO 2) = {1, 2} e Adj(BLOCO 3) = {2}. Suponhamos que deseja-se descobrir os blocos candidatos ao agrupamento ao ponto de articulação 1. Como Adj(BLOCO 1) – 1 = {}, e {} ⊂ Adj(1), o BLOCO 1 é candidato ao agrupamento. Temos ainda que Adj(BLOCO 2) – 1 = {2}; como {2} ⊂ Adj(1) então o BLOCO 2 também é candidato ao agrupamento e também o BLOCO 3 seguindo o mesmo critério. Tendo definido os blocos candidatos, resta somente definir qual o critério de seleção do bloco a ser agrupado.

O critério de agrupamento consiste na verificação do grau de proximidade do ponto de articulação com os blocos com que este se relaciona. Sendo assim, podemos

identificar inicialmente duas variáveis, que seriam o grau de entrada do nodo (*indegree*) e o grau de saída (*outdegree*) com relação apenas ao bloco analisado, e não o grau total no grafo.

Portanto, o grau de proximidade P entre um bloco b_n e um ponto de articulação v será dado através de:

$$P(v, b_n) = outdegree(v, b_n) - indegree(v, b_n)$$

onde $outdegree(b_n)$ corresponde ao grau de saída do ponto de articulação com relação ao bloco b_n e $indegree(b_n)$ corresponde ao grau de entrada do ponto de articulação com relação ao bloco selecionado.

Dado um ponto de articulação relacionado a um número variado de blocos, o agrupamento será realizado junto ao(s) bloco(s) com o maior valor de proximidade, ou seja, aquele que tiver o maior valor $P(v, b_n)$. Caso exista mais de um bloco com maior valor de proximidade, como critério de seleção, consideram-se as listas de adjacências do ponto de articulação e dos blocos analisados. Digamos que com o grafo da Figura 5.10 estejamos definindo qual bloco será agrupado ao ponto de articulação 1. Temos as saídas do ponto de articulação com relação ao BLOCO 1 em azul e as entradas no ponto de articulação provenientes do mesmo em vermelho. Com relação ao BLOCO 2, temos as saídas em verde e as entradas em roxo. Portanto, no exemplo dado, temos que:

$$P\ (1, \text{BLOCO } 1) = 1 - 3 = -2$$

$$P\ (1, \text{BLOCO } 2) = 1 - 1 = 0$$

Como P (1, BLOCO 2) > P (1, BLOCO 2), temos que o ponto de articulação 1 será agrupado ao BLOCO 2. Sendo assim, todos os pontos de articulação subseqüentes passarão por este processo, sendo que um bloco pode ser agrupado a um único ponto de articulação. Uma vez agrupado ele é excluído da lista de candidatos.

Finalmente, após a execução destas etapas, espera-se que o resultado obtido seja apropriado para a compreensão da informação, transmitindo de maneira mais fiel, o contexto da informação, assim como a estrutura de navegação geral existente entre os agrupamentos localizados. Vale ressaltar que na etapa de visualização os nodos isolados são inclusos novamente. A Figura 5.11 apresenta o resultado obtido após a execução do processo descrito acima e a Figura 5.12 apresenta o tutorial da Figura 5.4 considerando o agrupamento dos sub-grafos e respectiva representação na forma de um único nodo.

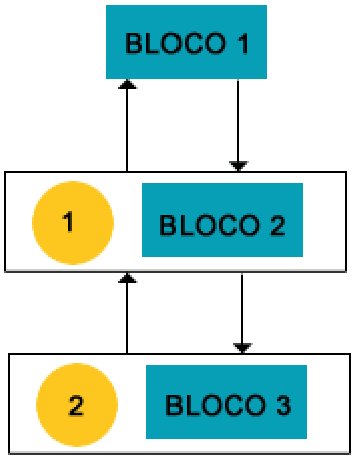


FIGURA 5.12 - Resultado do agrupamento da estrutura da Figura 5.9

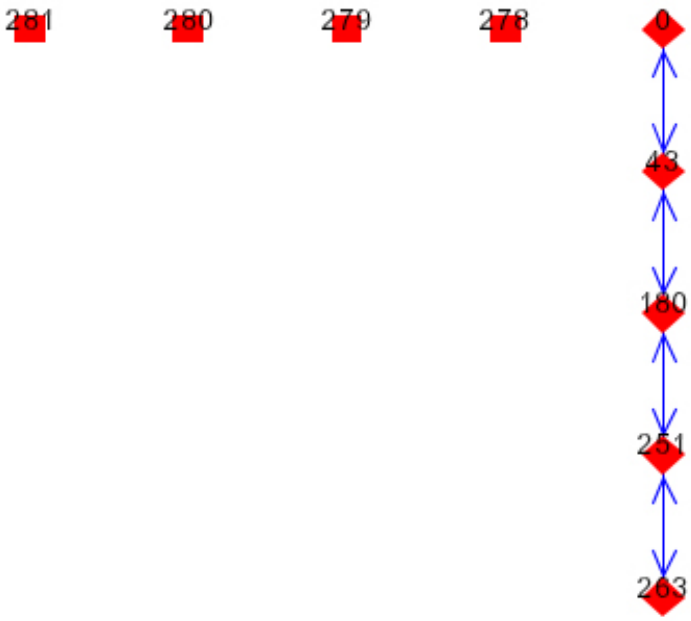


FIGURA 5.13 - Tutorial da Figura 5.4 agrupado: visualização gerada com a ferramenta Royere [MAR 2000]

5.4.2 Expansão dos agrupamentos

Uma vez apresentado o autômato com agrupamento, deve ser possível a um usuário obter uma visualização da "explosão" dos agrupamentos, assim como a visualização das saídas relacionadas.

Sendo assim, foram geradas as visualizações dos nodos dos sub-grafos sob duas perspectivas: a primeira delas, considerando apenas os relacionamentos contidos dentro

de um grafo, e a segunda considerando os relacionamentos dos nodos de um agrupamento com nodos pertencentes a outros agrupamentos.

A Figura 5.13 e a Figura 5.14 apresentam as visualizações geradas num contexto onde a quantidade de nodos é pequena, e outra onde a quantidade de nodos é grande, respectivamente.

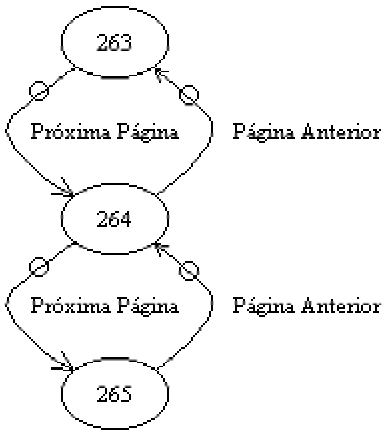


FIGURA 5.14 - Expansão de um agrupamento com poucos estados e sem relacionamentos inter-agrupamento

Já as Figuras 5.15 e 5.16 apresentam os resultados obtidos para agrupamentos com poucos nodos e agrupamentos relativamente grandes, considerando a apresentação dos relacionamentos inter-agrupamento, ou seja, relacionamentos entre nodos pertencentes a agrupamentos diferentes. Para esta opção foi utilizada a técnica de coloração de maneira que fossem destacados os nodos de outros agrupamentos, assim como os relacionamentos entre estes nodos e os nodos do agrupamento de interesse.

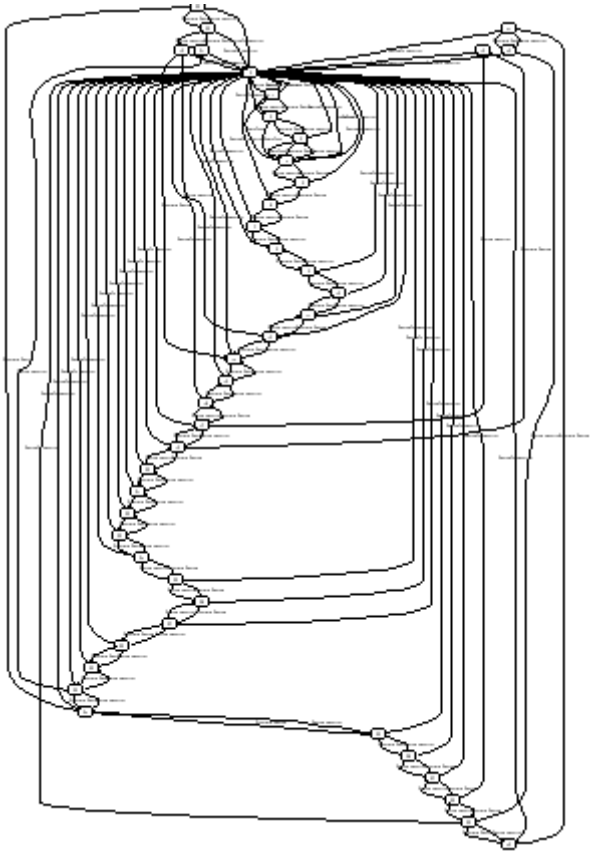


FIGURA 5.15 - Expansão de um agrupamento com muitos nodos e sem relacionamentos inter-agrupamento

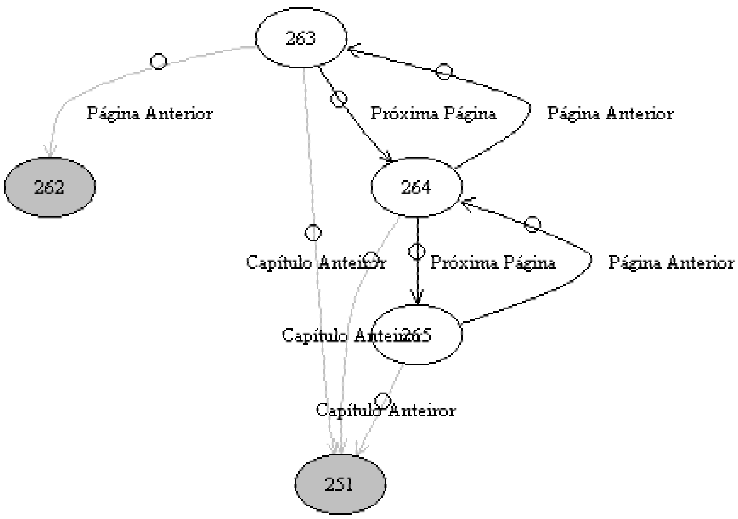


FIGURA 5.16 - Explosão de um agrupamento com poucos estados e relacionamentos inter-agrupamento

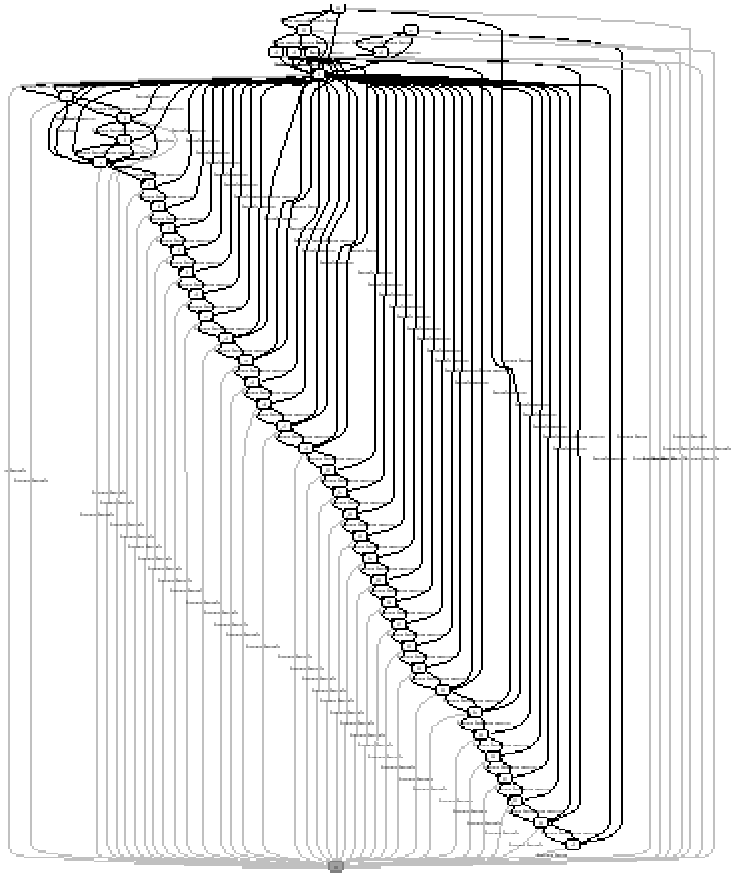


FIGURA 5.17 - Explosão de um agrupamento com muitos nodos e relacionamentos inter-agrupamento

Através das visualizações geradas, pode-se dizer que o layout obtido pela ferramenta é suficiente para a visualização de agrupamentos com poucos nodos. Porém, ao trabalhar com quantidades de elementos maiores, perde-se a efetividade da técnica. Como na primeira análise, quando todo o autômato foi visualizado, o mesmo problema é encontrado na visualização do agrupamento, apesar da relativa melhora na visibilidade da informação.

Faz-se necessário então o uso de técnicas de visualização que possibilitem a visualização das estruturas de navegação, assim como a visualização dos detalhes relacionados aos estados. Neste aspecto, as técnicas do tipo foco+contexto, que favorecem a visualização de detalhes sem a perda do contexto de visualização, tornam-se uma alternativa interessante.

5.4.3 Visão olho-de-peixe sobre autômatos finitos com saída

Após a aplicação da visualização dos autômatos finitos com estados agrupados, os seguintes requisitos devem ser atingidos:

- compreensão do contexto da informação;
- visualização da estrutura de navegação;

- visualização das informações de cada estado (ou seja, saídas relacionadas);

- visualização das informações relacionadas às saídas.

Como foi dito anteriormente, as técnicas foco+contexto mostraram-se as mais indicadas uma vez que agregam dois tipos de visualização: uma global, que envolve todo o grafo e outra detalhada, de um nodo específico. Ou seja, o nodo em foco é mostrado em mais detalhe que os demais nodos, mas sem que as demais informações deixem de ser visualizadas para que não se perca o contexto global da informação. Vale lembrar ainda que, na visualização das saídas, é importante demonstrar a ordem em que as mesmas estão organizadas, pois não existem relacionamentos entre elas; existem apenas informações relacionadas como o nome do arquivo representado.

Sendo assim, duas opções são levantadas para a solução desta questão: a utilização do layout hiperbólico ou a do layout tipo olho-de-peixe. O layout hiperbólico permite a implementação tanto em 2D como em 3D, porém é mais comumente aplicado a estruturas hierárquicas do tipo de árvores. Os fundamentos geométricos dessa técnica não são baseados em funções de distância claramente explicitadas, mas em mapeamento baseado na geometria hiperbólica.

A visão olho-de-peixe, por sua vez, apesar de possuir características que também favoreçam a visualização de estruturas hierárquicas, é aplicável a variados tipos de estruturas.

Devido à facilidade de utilização dos conceitos geométricos utilizados e explicitados por Sarkar e Brown [SAR 92], optou-se pela realização de uma extensão da visão olho-de-peixe para que esta pudesse suportar a visualização de autômatos finitos com saída. Seguindo a linha proposta de outros trabalhos encontrados na literatura, como [CAV 2000], a adaptação a ser realizada deverá suportar dois focos:

- foco do contexto da informação, ou seja, o estado selecionado;

- foco do detalhe, ou seja, foco das saídas selecionadas.

A Figura 5.17 apresenta o modelo projetado para exibição das informações nesta visualização. Trata-se de uma espécie de visão olho-de-peixe aninhada, já que a técnica é aplicada para a visualização do contexto da informação e, em seguida, para a visualização dos detalhes das saídas do nodo em foco.

Vale ressaltar que o esquema apresentado é um modelo gráfico gerado apenas com o objetivo de demonstrar qual o resultado final a ser obtido. Não se trata ainda do resultado obtido através da implementação das funcionalidades da ferramenta, e sim uma esquematização do objetivo a ser atingido. Os nodos são redimensionados de acordo com os valores da função de DOI (calculada em função da API e da distância), sendo que o mesmo conceito deverá ser aplicado à visualização das saídas relacionadas a cada um deles.

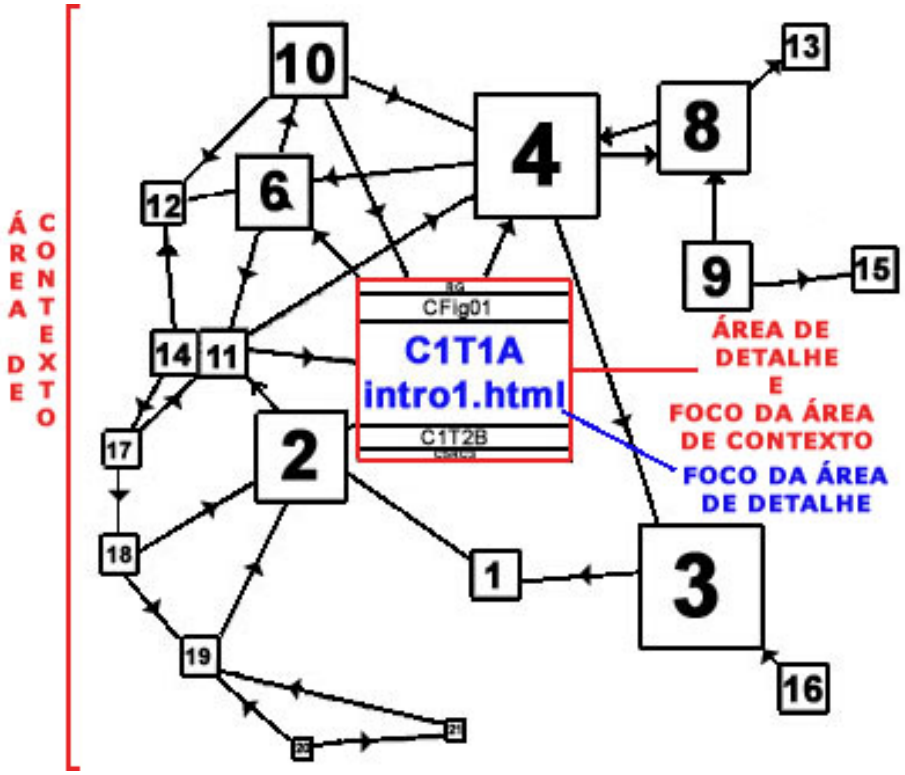


FIGURA 5.18 - Modelo da técnica proposta

Para a visualização do contexto da informação, foram utilizadas as funções definidas por Sarkar e Brown [SAR 92]. Essas funções se mostraram bastante apropriadas, uma vez que as transformações não são aplicadas a todos elementos do grafo. Sendo assim, a técnica não implica num custo computacional muito grande para a transformação das arestas, já que estas não são deformadas. O redimensionamento dos nodos se dá em função do tamanho normal do nodo, sua posição nas coordenadas originais, a posição do foco nas coordenadas originais e valor de sua função API (*A Priori Interest*).

A técnica apresentada é aplicada considerando que já existe um layout da visualização que está sendo transformado, ou seja, são conhecidas as coordenadas de cada nodo e o tamanho apresentado por eles. As funções são definidas sendo idênticas àquelas apresentadas no trabalho de Sarkar e Brown. *Sfeye(v, fp)* corresponde ao tamanho de um nodo *v*, com relação a um ponto de foco *fp*.

$$Sfeye\ (v, fp) = f\ (Snorm(v), Pnorm(v), Pnorm(fp), API(v)) \qquad (1)$$

Já a função que define o tamanho final do nodo na visão olho-de-peixe é definida em Sarkar como:

$$Sfeye\ (v, fp) = Sgeom(c.API(v))^{e} \qquad (2)$$

Aqui, teremos o tamanho definido como:

$$Qfeye\ (v) = centro(v) + s.Snorm(v)/2 \qquad (3)$$

$$Sgeom\ (v) = 2min(|Qfeye_x - Pfeye_x|, |Qfeye_y - Pfeye_y|) \qquad (4)$$

Qfeye(v) corresponde a um ponto localizado a uma distância *s.Snorm(v)/2* do centro do vértice e afastando-se do foco. *Snorm(v)* corresponde ao tamanho do nodo *v* na visão normal, ou seja, o tamanho utilizado no layout inicial, da mesma forma que *Pnorm*(v) também se refere às coordenadas do mesmo nodo *v* no layout normal. *Sgeom* é o tamanho geométrico do nodo, *s* é o fator de escala a ser aplicado, *centro* é a função que calcula o ponto central do nodo, *c* e *e* são constantes e *fp* é o nodo em foco. Neste trabalho, no lugar da função definida por Sarkar, o tamanho final do nodo na função olho-de-peixe foi definido como:

$$Sfeye(v, fp) = (Sgeom/Snorm(v))*API \quad (5)$$

onde o valor de *Sgeom* é calculado de acordo com as equações (3) e (4) definidas anteriormente. Os nodos são redimensionados de acordo com o grau de importância (API), sendo ainda utilizado um critério, que seria o valor de VW (*visual worth* ou importância visual) mínimo para a exibição do nodo sendo β e γ, constantes definidas pelo usuário.

$$VW (v, fp) = \beta\ Sfeye (v, fp) + \gamma \quad (6)$$

Já a função de grau de interesse, que foi definida inicialmente por Furnas como uma função do valor de API do nodo juntamente com a distância, pode ser definida como

$$DOI_{context} (v, fp) = API (v) – D (v, fp) \quad (7)$$

sendo a função API o valor de grau de importância de um nodo independente do foco em questão.

Para a visualização do nodo em foco, além da técnica proposta por Sarkar e Brown, deve ser aplicado um recurso adicional, para que assim se permita a visualização dos nodos relacionados a este elemento. Bartram *et al.* [BAR 95] desenvolveram um método baseado na visão olho-de-peixe chamado de *Continuous Zoom*. A técnica apresenta algumas características interessantes e relevantes, como o suporte a múltiplos focos e à manipulação independente dos eixos X e Y. Este trabalho independente com os eixos de coordenadas se dá pelo fato de o método realizar todas as transformações baseando-se na alocação global de espaço para seções retangulares do display, ou seja, projeções dos limites dos nodos nos eixos X e Y. Estas projeções, por sua vez, são utilizadas juntamente com fatores de escala e, em seguida, ajustadas de acordo com o grau de interesse, para assim, refletir as alterações desejadas no display.

Para possibilitar a aplicação da técnica proposta à visualização das saídas dentro do nodo, foram realizadas certas adaptações, uma vez que o espaço a ser considerado se trata do espaço alocado para o nodo em foco e que os intervalos entre os nodos são inexistentes. Além disso, como as alterações visam apenas a distribuição do espaço total do nodo entre as saídas de acordo com a saída em foco, é eliminada a necessidade de cálculos para o eixo X.

O tamanho requerido para a visualização da saída em foco (Eq. 8) é calculado em função do seu tamanho normal *Lnorm* (Eq. 9), caso não fosse aplicada à técnica de ênfase visual sobre a saída, juntamente com o fator de escala *s*, aplicado anteriormente

para a visualização do grafo como um todo e do fator de escala *Ss* a ser aplicado na visualização da saída em foco.

$$Lreq\ (sf) = Lnorm^{*}\ (s/Ss) \quad (8)$$

$$Lnorm = Sfeye(v, fp)/nOutputs \quad (9)$$

Sfeye é o tamanho do nodo na visão olho-de-peixe, dado pela Eq. 1, e *nOutputs* é o total de saídas relacionadas ao nodo. Esta informação pode ser extraída dos arquivos descritores. Para que as saídas sejam apresentadas mantendo as coordenadas coerentes com o nodo ao qual elas pertencem, as coordenadas são calculadas pela função *Psnorm*, pois a única coordenada disponível é a do nodo que está em foco. Considerando que a coordenada apresentada corresponde ao centro do nodo em questão, tem-se que:

$$Psnorm(si, fp) = Pfeye(fp) - dif/2 + i*Lnorm \quad (10)$$

onde *Pfeye(fp)* corresponde à posição do nodo ao qual a saída pertence, *dif/2* corresponde à diferença entre os tamanhos do nodo em relação ao tamanho normal da saída, lembrando que somente é necessário o trabalho com as coordenadas no eixo Y. *i* corresponde à ordem na qual o nodo é armazenado na estrutura. Esta ordem é a mesma apresentada nos arquivos descritores tendo uma sequência que vai de 0 a *n*.

Como na visualização dos nodos, a visualização das saídas também requer a aplicação de uma função de grau de interesse, ou seja, o grau de interesse de uma saída na visualização é calculado conforme a saída selecionada como foco. Uma função de grau de interesse baseada na distância das coordenadas das saídas em relação às coordenadas do foco já se mostra suficiente para o objetivo visado (Eq. 11).

$$DOI(output, sf) = 1/-D(output, sf) \quad (11)$$

Tendo calculado os valores anteriores, utiliza-se uma função chamada *doifi*, que se trata de uma função de grau de interesse recalculada (Eq. 12) e que deverá ser utilizada no cálculo da quantidade de espaço, *outputSize*, da área de detalhe a ser atribuída a cada uma das saídas presentes no nodo (Eq. 14), sendo que o espaço é calculado tendo como base a quantidade de espaço livre *freeSpace* (Eq. 13) disponível para a visualização da saída.

$$doifi\ (output) = DOI(output)/\sum D(output, sf) \quad (12)$$

$$freeSpace = Sfeye - reqSize \quad (13)$$

$$outputSize(output) = doifi(output)^{*}\ freeSpace \quad (14)$$

Após a atribuição de espaço às saídas, deve-se recalcular a quantidade de espaço livre *freeSpace** (Eq. 15) ainda disponível para a visualização. Caso este valor seja maior que 0, ele deve ser redistribuído entre as saídas de forma a eliminar o espaço livre em sua totalidade. Aqui a distribuição foi realizada de forma igualitária, já que a maior parte do espaço já foi distribuída e pouca coisa restará para a distribuição. Sendo assim, caso haja a necessidade, o tamanho final da saída é recalculado e representado pelo valor de *outputSize** (Eq. 16).

$$freeSpace^{*} = freeSpace - \sum outputSize \quad (15)$$

$$outputSize^* = outputSize + freeSpace^*/nOutputs \qquad (16)$$

Finalmente, depois de efetuadas as transformações sobre os tamanhos dos nodos, deve-se realizar o reajuste das coordenadas para que as saídas sejam visualizadas de forma ordenada dentro dos nodos, ocupando o mesmo espaço antes a elas atribuído. Este reajuste se dá através da função *Psfeye* (Eq. 17), onde *Psfeye* é a nova posição da saída após a aplicação da visualização olho-de-peixe a elas.

$$Psfeye = Psnorm(si, v) + \sum (n-1)*dif + dif(n)/2 \qquad (17)$$

Aqui, *dif* é a diferença entre os tamanhos das saídas na visão original (ou normal) e na visão olho-de-peixe. Quanto ao nível de detalhe, somente serão exibidos maiores detalhes da saída que é o foco da visualização. As demais saídas terão somente seus rótulos exibidos. A Figura 5.18 mostra a visualização de um grafo através da implementação das funções apresentadas.

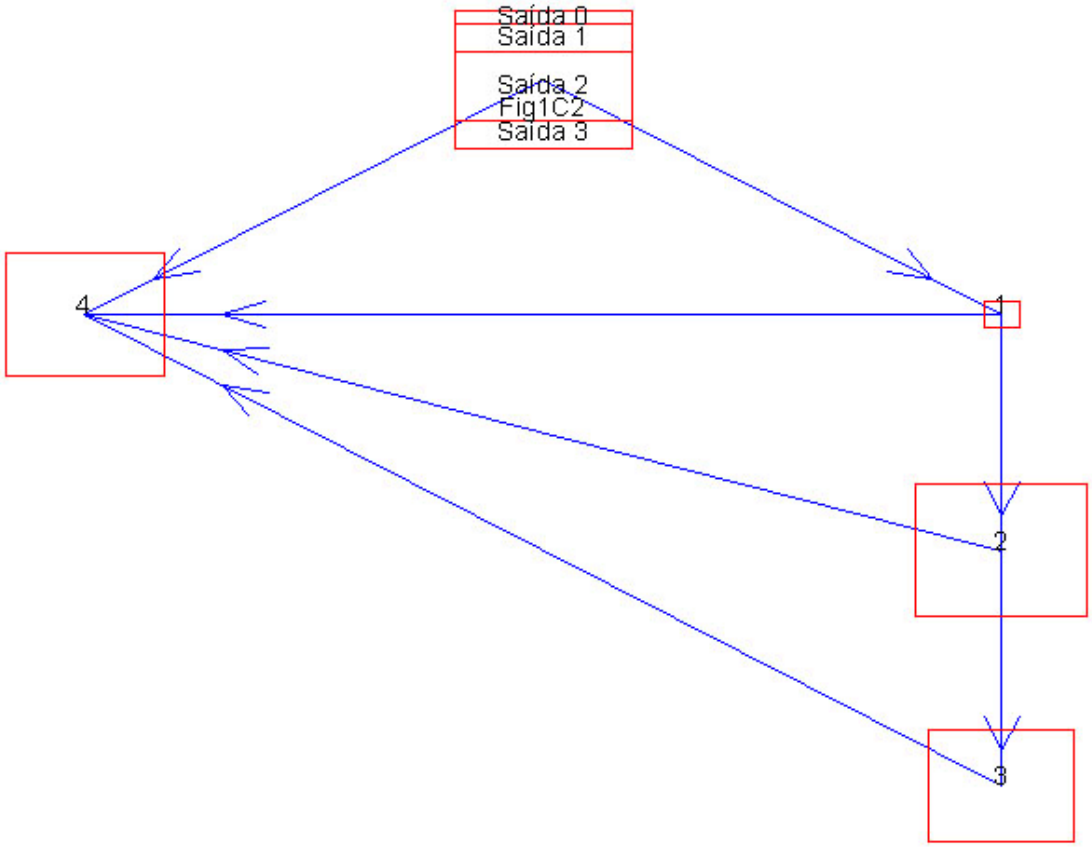


FIGURA 5.19 - Visualização dos autômatos finitos com saída utilizando a técnica proposta

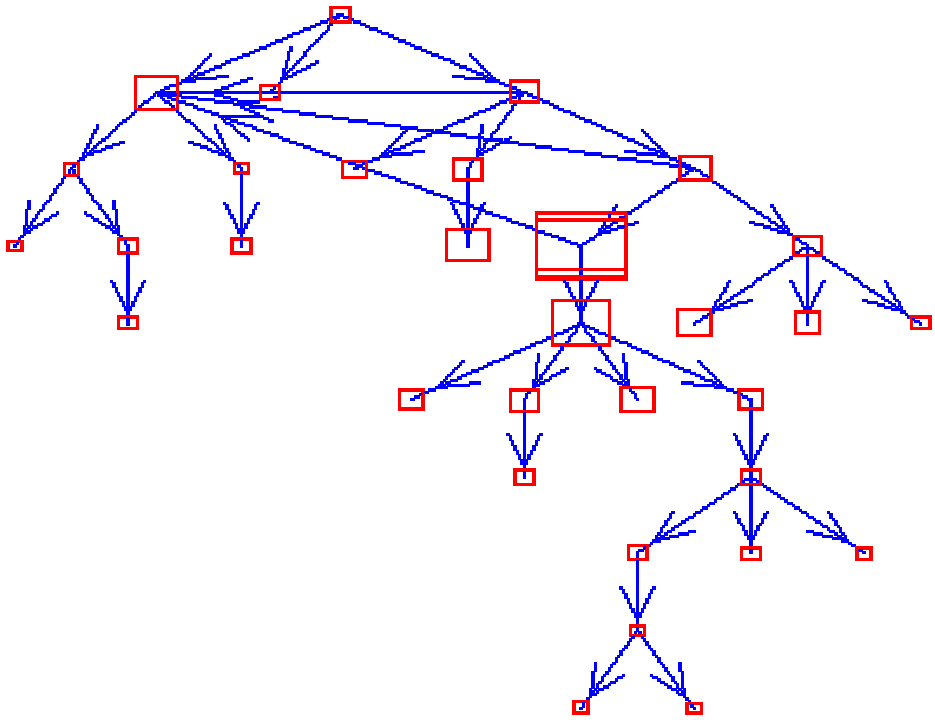


FIGURA 5.20 - Visualização de autômato finito com saída utilizando a ferramenta Royere [MAR 2000]

5.4.4 Extensões da ferramenta Royere

O processo de visualização se dá através de uma série de etapas que se iniciam com a seleção e armazenamento de todas as informações necessárias e contidas no Hyper-Automaton que seriam:

- Arquivos de saídas
- Arquivo do alfabeto de entrada;
- Programa descritor da estrutura do autômato a ser visualizado;

Para que as extensões feitas à DTD GraphXML tenham efeito na ferramenta e as informações das saídas inclusas na descrição do autômato possam ser visualizadas, mostrou-se necessária a realização de extensões às classes de parsers da ferramenta Royere. Sendo assim, as seguintes classes da ferramenta Royere foram alteradas:

- AbstractParser.java
- GraphSemantics.java
- GraphSemanticsAdapter.java

- InterpretDocument.java
- GraphXMLSemantics.java
- Keys.java
- GraphCsvMediator.java
- GraphDomMediator.java
- DebugGraphSemantics.java

Complementando as extensões dos parsers XML, para efetivar a visualização das estruturas descritas, foram realizadas extensões às seguintes classes:

- FisheyeControl.java – classe responsável pelo reposicionamento e redimensionamento dos nodos para a visão olho-de-peixe. Esta classe foi inclusa de forma a aplicar o mesmo conceito na visualização das saídas.
- DrawingControl.java – classe que contém as funções de controle para o desenho dos elementos de um grafo. Nesta classe foram inclusos as funções e cálculos específicos para a visualização da área de detalhe.
- ElementDrawing.java – classe que implementa funções específicas para o desenho dos elementos de um grafo. Foram inclusos portanto, métodos para permitir a geração da visualização das saídas de um nodo.

5.4.5 Generalização das técnicas desenvolvidas para estruturas de hiperdocumentos

Uma vez que a proposta apresentada não altera o layout obtido, e sim, uma transformação das posições e tamanhos dos elementos, a técnica desenvolvida é aplicável aos dois casos apresentados no início deste trabalho, ou seja, visualização das provas e exercícios adaptativos.

De maneira mais geral, apesar de direcionadas especificamente para a ferramenta Hyper-Automaton, as técnicas desenvolvidas neste trabalho são completamente aplicáveis a estruturas de hiperdocumentos em geral. Isto se deve justamente ao fato da base teórica do Hyper-Automaton ter sido construída visando o tratamento de hiperdocumentos. Como foi citado anteriormente, os hiperdocumentos podem perfeitamente ser representados por meio de grafos ou autômatos finitos. Sendo assim, a aplicação das técnicas de visualização de grafos desenvolvidas demostram ser viáveis para a representação de hiperdocumentos. .

No que se refere à visualização da explosão dos agrupamentos, a generalização se apresenta de uma forma simples. Assim como os estados possuem saídas, hiperdocumentos possuem elementos (tais como textos, imagens, animações e applets) relacionados a ele. Ou seja, assim como na visualização dos estados dos autômatos finitos era possível a representação das saídas relacionadas a um estado, na representação de um hiperdocumento, ao invés das saídas, poderão ser enumerados estes elementos relacionados ao documento, juntamente com os seus atributos.

Como exemplo do tipo de informação que pode ser apresentado no lugar das saídas, podem ser citadas as visualizações apresentadas pelas próprias ferramentas de edição existentes no mercado. Algumas delas já possibilitam uma visualização da estrutura de hiperdocumentos trabalhada, juntamente com a enumeração dos elementos agregados a cada documento. As Figuras 5.18 e 5.19 são exemplos de visualização gerados por uma ferramenta de edição.

Desta forma, para a efetiva aplicação da técnica, basta que sejam desenvolvidas metodologias eficazes para a extração de informações completas sobre estes elementos, tais como tamanho do arquivo (no caso de imagens ou outros arquivos anexados ao hiperdocumento) e parâmetros de entrada (no caso de scripts e applets), entre outros.

A técnica de agrupamento proposta pode ser aplicada de forma idêntica a hiperdocumentos em geral, sendo que obterá resultados mais efetivos quando aplicada a hiperdocumentos bem estruturados, principalmente no que se refere à navegabilidade da estrutura. No entanto, sua aplicação não se restringe apenas a este estudo de caso, pelo fato de utilizar apenas conceitos estruturais, pode ser aplicado a qualquer representação que utilize o formalismo de grafos. Neste caso, será apresentado um agrupamento dos blocos de um grafo junto aos seus pontos de articulação mais próximos.

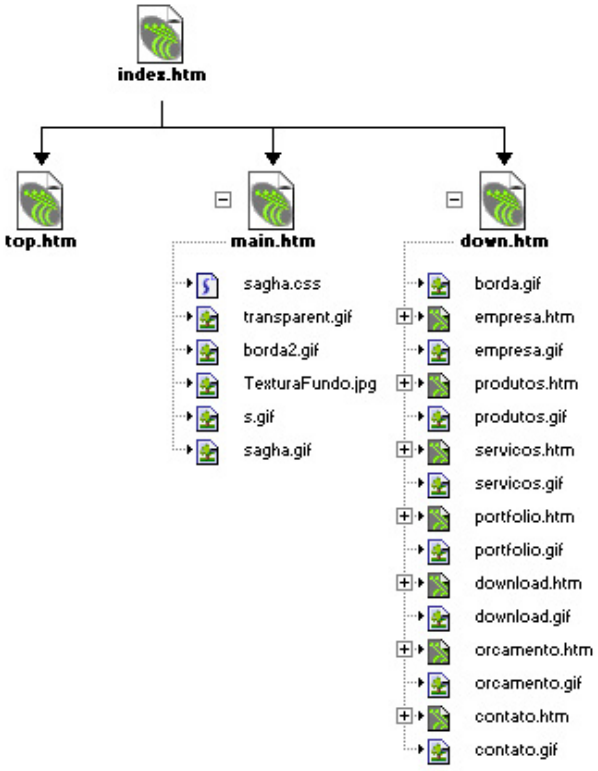


FIGURA 5.21 - Visualização de hiperdocumento[2] gerada pela ferramenta de edição de hiperdocumentos Macromedia Dreamweaver

[2] Site disponível em http://www.sagha.com.br

6 Conclusões

6.1 Avaliação geral do trabalho

A criação e o gerenciamento de estruturas de hiperdocumentos de grande complexidade é uma tarefa que exige muito cuidado e atenção do usuário. A aplicação de técnicas de Visualização de Informações apropriadas serve como método para imprimir maior eficácia à execução destas tarefas. Além disso, considerando que a WWW é um meio de publicação e divulgação consideravelmente barato e, cada vez mais acessível à maioria da população, tem-se que uma quantidade cada vez maior de usuários utilizando seus recursos.

Este trabalho apresentou uma proposta de visualização para complementar as facilidades oferecidas por uma ferramenta já desenvolvida, o Hyper-Automaton. Pretende-se que a técnica desenvolvida possa ser agregada à ferramenta em questão, imprimindo maior dinamismo a tarefas como gerenciamento e verificação dos dados.

As vantagens da utilização da técnica são inúmeras. A primeira delas trata-se da justificativa clássica da Visualização de Informações: a utilização da percepção visual contribui para a aceleração do processo de interpretação. O uso apropriado de recursos visuais pode aumentar sensivelmente a eficácia de um sistema, pela maior facilidade na compreensão das informações.

Com relação à visualização gerada em si, pode-se dizer que a aplicação de técnicas de agrupamento permite oferecer aos usuários (criadores dos hiperdocumentos) uma visão geral da estrutura desenvolvida além da navegação entre seus componentes. Reduz a complexidade visual da informação, além de melhorar o desempenho do sistema pelo fato de exigir a demonstração de uma quantidade reduzida de elementos.

A visualização dos autômatos finitos com saída, por sua vez, foi desenvolvida através da aplicação dos conceitos das técnicas foco+contexto, principalmente da visão olho-de-peixe. A técnica desenvolvida permite a visualização das estruturas das informações provendo duas áreas principais: a área de contexto e a área de detalhe.

A área de contexto apresenta a visão da estrutura de navegação, possuindo um foco que se destina à visualização de um nodo específico. Sendo assim, o nodo em foco é apresentado em maior detalhe, através da exibição de suas saídas, enquanto que os demais nodos são visualizados com maior ou menor destaque conforme sua distância do foco. O nodo em foco na área de contexto, por sua vez, constitui a área de detalhe, que é destinada à visualização dos detalhes das saídas relacionadas. Esta área de detalhe também possui uma região de foco, sendo que da mesma forma que a região de contexto, a saída em foco será apresentada com maior nível de detalhe e a exibição das demais saídas, com maior ou menor destaque, conforme a sua proximidade da saída em foco. Possibilita desta forma tanto a visualização das transições dos estados, assim como a visualização de detalhes específicos das saídas relacionadas ao estado selecionado como foco da visualização.

Além das características apresentadas, os níveis de escala a serem aplicados aos focos podem ser selecionados pelos usuários. Esta característica dá maior flexibilidade à

visualização, uma vez que é dada ao usuário a possibilidade de seleção do nível de escala que melhor se adapte às suas necessidades.

Técnicas do tipo foco+contexto como a que foi apresentada, apesar de desenvolvidas para visualização de autômatos finitos com saída, podem ser facilmente generalizadas sendo o caso de uso mais comum, o da estrutura de hiperdocumentos. No entanto, qualquer estrutura que possua elementos relacionados, assim como atributos relacionados a estes objetos pode se utilizar desta técnica.

6.2 Trabalhos Futuros

Como trabalhos futuros sugeridos podem ser citados:

- Ferramenta de edição de grafos: neste trabalho foram utilizados os recursos de algumas ferramentas de edição e visualização como o Dotty e o Royere. No entanto, uma ferramenta de edição de grafos que também agregue as necessidades do Hyper-Automaton seria mais apropriada.

- Técnicas para integração da visão geral com a visualização dos agrupamentos: as duas técnicas foram desenvolvidas separadamente, e ainda não foi realizada a integração das duas visões. Inicialmente, pode-se utilizar o *full-zoom* [SCH 98], que apresenta apenas detalhes do nível atual da hierarquia, ou seja, apresenta somente a visualização do agrupamento selecionado, sem considerar os demais agrupamentos existentes ou o contexto geral da informação. Entretanto, parece mais apropriado um estudo mais aprofundado de alternativas, antes da efetiva integração.

- Técnicas de visualização para estruturas de provas e exercícios adaptativos: apesar de aparentemente serem um problema de solução mais simples, como foi dito anteriormente, é necessário que haja um estudo sobre a técnica de layout mais apropriada, juntamente com a seleção de uma metodologia apropriada para a exibição dos detalhes dos nodos. Vale ressaltar que ainda estamos tratando de autômatos finitos com saída. É possível que a visão olho-de-peixe apresentada neste trabalho já seja suficiente, uma vez que permite a visualização foco+contexto das informações, considerando a existência de saídas. Mesmo assim, maiores estudos são necessários para comprovação da utilidade da técnica, ou mesmo para a seleção de um layout mais apropriado para cada tipo de estrutura, dadas suas particularidades.

- Estudo de técnicas de aprimoramento de layout: como em toda técnica de visualização, questões relacionadas à estética são de extrema importância. Sendo assim, estudos de técnicas relacionadas à colocação de rótulos ou desenho de arestas são de extrema importância. A colocação de rótulos se mostra importante, principalmente no que se refere à clareza visual, contribuindo para o melhor entendimento da informação por parte do usuário. Já a questão do cruzamento de arestas, se refere ao estudo do aprimoramento de técnicas de layout de forma a evitar a sua ocorrência. O cruzamento de arestas, assim como ocorreu neste trabalho, tem se mostrado como um dos principais agentes causadores de confusão visual.

- Implementação de métodos que permitam a visualização de grafos cíclicos na ferramenta utilizada. Como a ferramenta atualmente não permite a visualização de grafos cíclicos, a solução deste aspecto resultará num resultado mais efetivo para os usuários no que se refere à visualização das estruturas de navegação.

Referências

[AND 98] ANDERSSON, E. **Automatic Layout of Diagrams in Rational Rose**. Sweden: Uppsala University, 1998.

[BAK 96] BAKER, J.E et al. Algorithm Animation Over the World Wide Web**.** In: WORKSHOP ON ADVANCED INTERFACES, 1996. **Proceedings**… Disponível em:<http://citeseer.nj.nec.com/baker96algorithm.html>. Acesso em: 17 maio 2001.

[BAR 95] BARTRAM, L. et al. The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Spaces. In: ACM USER INTERFACE AND SOFTWARE TECHNOLOGY, UIST, 1995. Pittsburgh, USA. **Proceedings…** [S.l: s.n.], 1995. p.207-215. Disponível em: <http://www.cs.sfu.ca/~lyn/personal/Publications/uist95.pdf>. Acesso em 05 out. 2001.

[BAT 94] BATTISTA, G.Di et al. **Algorithms for Drawing Graphs:** an Annotated Bibliography**.** 1994. Disponível em: <http://www.cs.brown.edu/~rt/papers/gdbiblio.pdf>. Acesso em: 25 fev. 2001.

[BED 99] BEDERSON, B.B.; BOLTMAN, A. Does Animation Help Users Build Metal Maps of Spatial Information? In: IEEE INFORMATION VISUALIZATION SYMPOSIUM, 1999, San Francisco. **Proceedings…** Los Alamitos: IEEE Computer Society, 1999. p.28-35.

[BEK 2002] BERKHIN, P. **Survey of Clustering Data Mining Techniques.** Accrue Software, 2002. Disponível em: <http://www.accrue.com/products/rp_cluster_review.pdf>. Acesso em: 13 nov. 2002.

[BER 99] BERTAULT, F. A Force-Directed Method Algoritm that Preserves Edge Crossing Properties. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 7., 1999, Shring Castle. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1999. p. 351-358. (Lecture Notes in Computer Science, v. 1731).

[BIE 97] BIEDL, T. et al. Orthogonal 3-D Graph Drawing. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 5., 1997, Rome, Italy. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1997. p. 76-86. (Lecture Notes in Computer Science, v. 1353).

[BIE 98] BIEDL, T. et al. Graph Multidrawing: Finding Nice Drawings Without Defining Nice. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 6., 1998., Montreal, Canadá. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1998. p. 347-355 (Lecture Notes

in Computer Science, v. 1547).

[BJÖ 2001] BJÖRK, S. The ScrollSearcher Technique: Using Scrollbars to Explore Search Results. In: INTERNAL CONFERENCE ON HUMAN-COMPUTER INTERACTION, 2001, New Orleans. Disponível em: <http://www.viktoria.se/groups/play/publications/2001/scrollsearcher.pdf>. Acesso em: 07 dez. 2002.

[BOL 98] BOLEY, D. **Hierarchical Taxonomies Using Divisive Partitioning.** Minnesota: Minnesota: Department of Computer Science, University of Minnesota, 1998. Disponível em: <http://ftp.cs.umn.edu/dept/users/boley/reports/taxonomy.ps.gz>. Acesso em: 22 jun. 2002.

[BRI 97] BRIDGEMAN, S.S. et al. InteractiveGiotto: An Algorithm for Interactive Orthogonal Graph Drawing. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 5., 1997, Rome, Italy. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1997. p. 303-308 (Lecture Notes in Computer Science, v. 1353).

[BRI 99] BRIDGEMAN, S.; GARG, A.; TAMASSIA, R. A Graph Drawing and Translation Service on the WWW. 1999. Disponível em: <http://loki.cs.brown.edu:8081/graphserver/home.html>. Acesso em: 27 abr. 2002.

[CAD 99] CARD, S.K.; MACKINLAY, J.D.; SHNEIDERMAN, B. **Readings in Information Visualization - Using Visualization to Think.** San Francisco: Morgan Kaufmann, 1999. p. 307-330.

[CAR 93] CARRIERE, J.; KAZMAN, R. **Interacting with Huge Hierarchies:** beyond cone trees. 1993. Disponível em: <ftp://ftp.lri.fr/LRI/articles/mbl/ehci95/Carriere.ps.Z>. Acesso em: 17 jan. 2002.

[CAV 00] CAVA, R.A.; FREITAS, C.M.D.S. Visualizing Hierarchies Using a Modified Focus+Context Technique. In: IEEE SYMPOSIUM ON INFORMATION VISUALIZATION 2001, InfoVis 2001. Disponível em: <http://www.inf.ufrgs.br/~carla/papers/CavaFreitas-Infovis.pdf>. Acesso em: 14 nov. 2001.

[CHU 95] CHURCHER, N. **Applications of Distortion-Oriented Presentation Techniques in GIS.** Apresentado no AURISA/SIRC'95 – 7º Colloquium of the Spatial Information Research Center, 1995. Disponível em: <http://www.cosc.cantebury.ac.nz/research/reports/TechReps/1995/tr_9501.pdf>. Acesso em: 17 abr. 2002

[COH 95] COHEN, R.F. et al. Dynamic Graph Drawing: Trees, Series-Parallel Digraphs, and Planar St-Digraphs. **SIAM Journal on Computing**, Philadelphia, v. 24, n. 5, p. 970-1001, 1995. Disponível em: <http://www.cs.brown.edu/cgc/papers/cdtt-dgdts-95.os.gz>. Acesso em:

27 out. 2002

[CON 87] CONKLIN, J. Hypertext: an Introduction and Survey. **Computer**, Los Alamitos, v. 20, n. 9, p. 17-41, 1987.

[DOG 02] DOGRUSOZ, U.; FENG, Q.; MADDEN, B.; DOORLEY, M.; FRICK, A. Graph Visualization Toolkits. **IEEE Computer Graphics and Applications**, Los Alamitos, v. 22, n. 1, p. 30-37, 2002.

[EAD 96] EADES, P.; FENG. Q.-W. Multilevel Visualization of Clustered Graphs. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 4., 1996, Berkeley, California. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1996. (Lecture Notes in Computer Science, Vol. 1190). Disponível em: <http://www.reasearch.att.com/conf/gd96/p2/feng2.ps>. Acesso em: 11 out. 2002.

[FAS 99] FASULO, D. **An Analysis of Recent Work on Clustering Algorithms.** Seattle: University of Washington, 1999. Technical Report. Disponível em: <http://www.cs.washington.edu/homes/dfasulo/clustering.ps.>. Acesso em: 25 maio 2002.

[FOß 97] FÖßMEIER, U. Interactive Orthogonal Graph Drawing: Algorithms and Bounds. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 5., 1997, Rome, Italy. **Graph Drawing:** proceedings. Berlin: Springer Verlag, 1997. p. 111-123 (Lecture Notes in Computer Science, v. 1353).

[FRO 94] FRÖHLICH, M.; WERNER, M. **The Graph Visualization System daVinci – A User Interface for Applications.** Germany: University of Bremen, 1994. (Technical Report, n. 5/94). Disponível em: <ftp://ftp.tzi.DE/tzi/biss/daVinci/papers/techrep0594.ps.gz.>. Acesso em: 11 Ago. 2001.

[FRO 95] FRÖHLICH, M.; WERNER, M. Demonstration of the interactive Graph Visualization System da Vinci. DIMACS WORKSHOP ON GRAPH DRAWING, GD, 2., 1995, New Jersey. **Graph Drawing:** proceedings. Berlin: Springer-Verlag. p.266-269. (Lecture Notes in Computer Science v. 894)

[FRU 91] FRUCHTERMAN, T.M.J; REINGOLD, E.M. Graph-Drawing by Force-Directed Placement. **Software – Practice and Experience**, London, v. 21, n. 11, p. 1129-1164, Nov. 1991.

[FUR 81] FURNAS, G.W. **The FISHEYE view:** a new look at structured files. Bell Laboratories, 1981. Disponível em: <http://www.si.umich.edu/~furnas/Papers/FisheyeOriginalTM.pdf>. Acesso em: 06 maio 2001.

[GAN 2000] GANSNER, E.R.; NORTH, S.C. An Open Graph Visualization System and its Applications to Software Engineering. **Software – Practice and**

Experience, London, v. 30, n. 11, p. 1203-1233, Sept. 2000.

[GAR 94] GARG, A.; TAMASSIA, R. Advances in Graph Drawing. In: ALGORITHMS AND COMPLEXITY SECOND ITALIAN CONFERENCE, CIAC, 1994, Rome, Italy. **Algorithms and Complexity:** proceedings. Berlin: Springer-Verlag, 1994. p. 12-21 (Lecture Notes in Computer Science, v. 778)

[GER 97] GERSHON, N.; EICK, S. G. Information Visualization, **IEEE Computer Graphics and Applications**, Los Alamitos, v. 17, n. 4, p. 29-31, July/Aug. 1997.

[HE 98] HE, W.; MARRIOT, K. Constrained Graph Layout. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 4., 1996, Berkeley, California. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1996. p. 217-232. (Lecture Notes in Computer Science, v.. 1190).

[HER 99] HERMAN, I. et al. **Latour – a Tree Visualization System.** 1999. Disponível em: <http://www.cwi.nl/InfoVisu/papers/LatourOverview.pdf>. Acesso em: 14 set. 2001

[HER 2000] HERMAN, I. et al. **Graph Visualization and Navigation in Information Visualization:** a Survey**.** 2000. Disponível em: <http://www.cwi.nl/InfoVisu/Survey/StarGraphVisuInInfoVis.pdf>. Acesso em: 14 set. 2001.

[HER 2000b] HERMAN, I.; MARSHALL, M. S. GraphXML — An XML-based graph description format. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 8., 2000. Colonial Williamsburg, USA. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 2000. p. 52-62. (Lecture Notes in Computer Science, v. 1984). Disponível em: <http://www.cwi.nl/InfoVisu/GraphXML/GraphXMLShort.pdf>. Acesso em: 29 ago. 2002

[HIM 96] HIMSOLT, M. The Graphlet System (System Demonstration). In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 4., 1996, Berkeley, California. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1996. p. 233-240. (Lecture Notes in Computer Science, v. 1190). Disponível em: <http://www.research.att.com/conf/gd96/p2/himsolt.ps>. Acesso em: 22 dez. 2001.

[HIM 96b] HIMSOLT, M. **GML:** A portable graph file format. Disponível em: <http://infosun.fmi.uni-passau.de/Graphlet/GML/gml-tr.html>. Acesso em: 14 ago. 2002.

[HOL 97] HOLMQUIST, L. E. Focus+Context Visualization with Flip Zooming and the Zoom Browser. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI, 1997. **Proceedings…** Disponível em: <http://www.viktoria.informatik.gu.se/groups/play/publications/97/leh-chi97.pdf >. Acesso em: 22 mai. 2002.

[HOR 2001] HORNBAEK, K.; BEDERSON, B.B; PLAISANT, C. Navigation Patterns and Usability of Overview+Detail Zoomable Interfaces With and Without Overview. **ACM Transactions on Computer-Human Interaction**, New York, v. 9, n. 4, p. 362-389, 2001.

[JAN 2002] JANECEK, P.; PU, P. A Framework for Designing Fisheye Views to Support Multiple Semantic Contexts. In: INTERNATIONAL CONFERENCE ON ADVANCED VISUAL INTERFACES, AVI, 2002. Disponível em: <http://hci.epfl.ch/website/publication-doc/AVI_final_bw.pdf>. Acesso em: 10 jan. 2003.

[LAM 95] LAMPING, J.; RAO, R.; PIROLLI, P. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI, 1995. Disponível em: <http://www.ramanarao.com/papers/startree-chi95.pdf>. Acesso em: 18 ago. 2002.

[LIS 99] LISITSYN, I.A.; KASYANOV, V.N. Higres – Visualization System for Clustered Graphs and Graph Algorithms. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 7., 1999, Shrin Castle, Czech Republic. **Graph Drawing:** proceedings. Berlin: Springer Verlag, 1999. p. 82-89. (Lecture Notes in Computer Science, v.1731).

[MAC 2000] MACHADO, J. H. P; MORAIS, C.T.Q.; MENEZES, P.B. ; REIS, R.A.L. Structuring Web Course Pages as Automata: Revising Concepts. In: RIAO, 2000. **Content-Based Multimedia Information Access:** conference proceedings. Paris: CID, 2000. v. 1, p. 150-159.

[MAR 2000] MARSHALL, M. S. **Methods and Tools for the Visualization and Navigation of Graphs**. 2000. Tese (doutorado), Universidade de Bordeaux.

[MCC 99] McCRICKARD, D. S.; CATRAMBONE, R. Beyond the Scrollbar: An Evolution and Evaluation of Alternative Navigation Techniques. In: SYMPOSIUM ON VISUAL LANGUAGES, Tokio, Japan, 1999. **Proceedings…** Disponível em: <http://cuisung.unige.ch/Visual/local/McCrickardCatrambone99.ps.gz>. Acesso em: 27 jul. 2002.

[MEN 96] MENDELZON, A. Visualizing the World Wide Web. In: ADVANCED USER INTERFACES, AVI, 1996, Italy. **Proceedings…** Disponível em: <ftp://ftp.db.toronto.edu/pub/papers/avi96.os.gz>. Acesso em: 7 jul.

2002.

[MEE 2000] MENEZES, P.B.; MACHADO, J.P. Hyper-Automaton: Hypertext Framework with Categorical Operations. In: SIMPÓSIO BRASILEIRO DE LINGUAGENS DE PROGRAMAÇÃO, SBPL, 2000. **Anais...** Recife: Centro de Informática da UFPE. 2000. p. 29-47.

[MUC 98] MUCHLUAT, D.C.; RODRIGUES, R.F.; SOARES, L.F. WWW Fisheye-View Graphical Browser. In: MULTIMEDIA MODELING CONFERENCE, 1998, Lausanne, Switzerland. **Proceedings…** [S.l.: s.n.], 1998. p. 80-89.

[MUK 95] MUKHERJEA, S.; FOLEY, J.D.; HUDSON, S. Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI, 1995. **Proceedings…** Disponível em: <http://www.acm.org/sigchi/chi95/Electronic/documents/papers/sm_body.htm>. Acesso em: 10 set. 2002.

[MUN 2000] MUNZNER, T. **Interactive Visualization of Large Graphs and Networks**. 2000. Tese (Doutorado) – Stanford University, Stanford. Disponível em: <http://graphics.stanford.edu/papers/munzner_thesis/all.onscreen.pdf>. Acesso em: 5 jun. 2002.

[MUN 98] MUNZNER, T. Drawing Large Graphs with H3Viewer and Site Manager. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 6., 1999, Montreal, Canada. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1998. p. 384-393. (Lecture Notes in Computer Science, v. 1547). Disponível em: <http://graphics.stanford.edu/papers/h3draw/gd98.pdf>. Acesso em: 12 ago. 2002.

[NOI 93] NOIK, E. G. Layout-independent Fisheye Views of Nested Graphs. In: SYMPOSIUM ON VISUAL LANGUAGE, VL, 1993, Bergen, Norway. **Proceedings...** Disponível em: <ftp://ftp.db.toronto.edu/pub/noik/vl93.ps.Z>. Acesso em: 3 maio 2002.

[NOR 94] NORTH, S.C.; KOUTSOFIOUS, E. Applications of Graph Visualization. In: GRAPHICS INTERFACE, Banff, Canada, 1994. **Proceedings…** Disponível em: <http://www.reasearch.att.com/sw/tools/graphviz/GI94.pdf>. Acesso em 3 maio 2002.

[NOR 96] NORTH, S.C. Incremental Layout in DynaDag. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 4., 1996, Berkeley, California. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1996. (Lecture Notes in Computer Science, v. 1190). Disponível em: <ftp://ftp.research.att.com/dist/drawdag/dynadag.os.gz>. Acesso em: 14 ago. 2002.

[PAP 97] PAPAKOSTAS, A.; TOLLIS, I.G. Incremental Orthogonal Graph Drawing in Three Dimensions. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 5., 1997, Rome, Italy. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1997. p. 52-63. (Lecture Notes in Computer Science, v. 1353).

[PLA 2002] PLAISANT, C.; GROSJEAN, J.; BEDERSON, B.B. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. IEEE SYMPOSIUM ON INFORMATION VISUALIZATION, InfoVis, 2002. **Proceedings…** Boston: CS Press, 2002. p. 57-64.

[PUR 98] PURCHASE, H. Wich Aesthetic Has the Greatest Effect on Human Understanding? In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 7., 1997, Rome, Italy. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1997. p. 248-261. (Lecture Notes in Computer Science, v. 1353).

[RIS 00] RISDEN, K. et al. An Initial Examination of Ease of Use for 2D and 3D Information Visualizations of Web Content. **International Journal of Human-Computer Studies**, [S.l.], v. 53, p. 695-714, 2000.

[RYA 96] RYALL, K.; MARKS, J.; SHIEBER, S. An Interactive System for Drawing Graphs. In: INTERNAL SYMPOSIUM ON GRAPH DRAWING, GD, 4., 1996, Berkeley, California. **Graph Drawing:** proceedings. Berlin: Springer-Verlag, 1996. p. 387-394. (Lecture Notes in Computer Science, v. 1190).

[SAR 92] SARKAR, M.; BROWN, M. H. Graphical Fisheye Views of Graphs. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI, 1992. **Proceedings…** Disponível em <ftp://gatepeeker.dec.com/pub/DEC/SRC/research-reports/SRC-084a.ps.gz>. Acesso em: 16 ago. 2002.

[SCH 98] SCHAFFER, D. et al. Navigating Hierarchically Clustered Networks Through Fisheye and Full-Zoom Methods. **ACM Transactions on Computer-Human Interaction**, New York, 1998. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers/1996/96-Fisheye.TOCHI/fisheyetochi96.pdf>. Acesso em: 07 out. 2002.

[STO 97] STOREY, M.–A.D. On Integrating Visualization Techniques for Effective Software Exploration. In: IEEE SYMPOSIUM ON INFORMATION VISUALIZATION, InfoVis, 1997. **Proceedings...** Disponível em: <http://www.cs.uvic.ca/~mstorey/papers/infovis97.pdf>. Acesso em: 16 ago. 2002.

[SZW 84] SZWARCFITER, J.L. **Grafos e algoritmos computacionais.** Rio de Janeiro: Campus, 1984. p. 52-56.

[TAM 97] TAMASSIA, R. Graph Drawing. In: **CRC Handbook of Discrete and Computational Geometry.** Boca Raton, FL: CRC Press,1997.

[TOL 96] TOLLIS, I.G. Graph Drawing and Information Visualization. **ACM Computing Surveys**, New York, v. 28, n. 19**,** 1996. Disponível em: <http://www.utdallas.edu/~tollis/SDCR96/TollisGeometry.html>. Acesso em: 27 nov. 2002.

[TÖL 00] TÖLLE, J.; NIGGEMANN, O. Supporting Intrusion Detection by Graph Clustering and Graph Drawing. In: INTERNATIONAL WORKSHOP ON RECENT ADVANCES IN INTRUSION DETECTION, RAID, 2000, Toulouse, France. **Proceedings...** Disponível em: <http://www.raid-Symposium.org/raid2000/Materials/Abstracts/22/22.pdf >. Acesso em: 05 fev. 2002.

[WIJ 99] WIJK, J.J. van; WETERING, H. van. Cushion Treemaps: Visualization of Hierarchical Information. In: IEEE SYMPOSIUM ON INFORMATION VISUALIZATION, InfoVis, 1999. **Proceedings...** Disponível em: <http://www.win.tue.nl/~vanwijk/ctm.pdf>. Acesso em: 13 abr. 2002.

[WIL 99] WILSON, R.M; BERGERON, R.D. Dynamic Hierarchy Specification and Visualization. In: IEEE SYMPOSIUM ON INFORMATION VISUALIZATION, InfoVis, 1999. **Proceedings...** California: IEEE CS Press, 1999. p. 65-72.