

INSTITUTO MILITAR DE ENGENHARIA
EXÉRCITO BRASILEIRO
SECRETARIA DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

WAGNER ANTONIO ARBEX

ASPECTOS DE VISUALIZAÇÃO DE REDES

Rio de Janeiro
2002

INSTITUTO MILITAR DE ENGENHARIA

WAGNER ANTONIO ARBEX

ASPECTOS DE VISUALIZAÇÃO DE REDES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientadora: Profa. Lilian Markenzon – D. Sc.

Rio de Janeiro
2002

INSTITUTO MILITAR DE ENGENHARIA

WAGNER ANTONIO ARBEX

ASPECTOS DE VISUALIZAÇÃO DE REDES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientadora: Profa. Lilian Markenzon – D. Sc.

Aprovada em 25 de julho de 2002 pela seguinte Banca Examinadora:

Profa. Lilian Markenzon – D. Sc. do IME – Presidente

Prof. Oswaldo Vernet de Souza Pires – D. Sc. do NCE/UFRJ

Profa. Cláudia Marcela Justel – D. Sc. do IME

Rio de Janeiro
2002

AGRADECIMENTOS

À professora Lílian Markenzon, orientadora deste trabalho, por dispor de seu tempo, pela dedicação e inúmeras propostas para que esta dissertação se tornasse objetiva e consistente.

Aos professores Cláudia Marcela Justel e Oswaldo Vernet de Souza Pires, membros da banca examinadora, pela seriedade, interesse e respeito que demonstraram na avaliação deste trabalho.

Ao Instituto Militar de Engenharia, ao seu corpo docente e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior pelas oportunidades e recursos oferecidos para a obtenção de um verdadeiro aprendizado.

Ao Centro de Ensino Superior de Juiz de Fora, que através de seu diretor, professor José Ventura, incentivou e apoiou incondicionalmente a realização deste trabalho.

Aos “Net Street Boys”, companheiros de todos os dias, aos meus pais, Josephina e Antonio Arbex, aos meus irmãos e respectivos “agregados”, meus sobrinhos, à Cristiana Borges, por estar ao meu lado e ser quem é, e em especial à minha filha Luna Arbex, pelo carinho com que sempre me cercaram e aos quais dedico este trabalho.

SUMÁRIO

AGRADECIMENTOS	3
LISTA DE ILUSTRAÇÕES	6
1 INTRODUÇÃO	13
1.1 O que é visualização	13
1.2 O que são redes de serviço.....	14
1.3 O que são softwares de visualização	17
1.4 Proposta de trabalho	18
2 TRABALHOS EXISTENTES NA LITERATURA	20
2.1 introdução e revisão bibliográfica	20
2.2 Trabalhos de maior destaque	31
2.3 Conclusões.....	42
3 O KINEGRAPH	46
3.1 Apresentação	46
3.2 Comentários e avaliação	48
4 O KITNET	51
4.1 Apresentação e objetivos	51
4.2 Descrição das plataformas e do protótipo	52
4.3 Entrada de dados da rede	57
4.4 Edição da rede	65
4.5 Visualização	68
4.6 Desenvolvimento	77
4.6.1 Leitura, gravação e descrição das estruturas de dados	77

4.6.2	Importação e exportação de dados	79
4.6.3	Especificação de outros detalhes relevantes	83
4.7	Comentários e avaliação	88
5	CONCLUSÕES E TRABALHOS FUTUROS	92
6	REFERÊNCIAS BIBLIOGRÁFICAS	95
7	ANEXOS	100
7.1	ANEXO 1: ALGUNS EXEMPLOS DE VISUALIZAÇÃO.....	101
7.2	ANEXO 2: ELEMENTOS E PROCEDIMENTOS ENVOLVIDOS NA VISUALIZAÇÃO DE REDES DE SERVIÇO	120

LISTA DE ILUSTRAÇÕES

FIG. 2.1	Expansão e contração de nós dentro da hierarquia (EICK & WILLS, 1993).	23
FIG. 2.2	Exemplo de uma vista possível em uma rede global hipotética (HE & EICK, 1998).	26
FIG. 2.3	Exemplo de visualização de uma rede com a execução animada de um algoritmo (COULLARD et al., 1999).	29
FIG. 2.4	Menu de opções de edição da rede (COULLARD et al., 1999).	30
FIG. 2.5	Etapas do modelo de visualização (CARMO & CUNHA, 1998) (ALVES et al., 2000).	37
FIG. 4.1	Menu de procedimentos.	55
FIG. 4.2	Menu “Rede”	55
FIG. 4.3	Menu “Ver”	56
FIG. 4.4	Menu “Ver”	56
FIG. 4.5	Tela inicial do KitNet.	58
FIG. 4.6	Menu “Rede” → “Criar...” - Tela para criação automática da rede.	59
FIG. 4.7	Exemplo da tela para criação de uma rede de forma automática.	60
FIG. 4.8	Legenda da tela de visualização.	61
FIG. 4.9	Visualização da instância 6 de REDE1.	62

FIG. 4.10	Visualização da instância 7 de REDE1.....	63
FIG. 4.11	Visualização da instância 8 de REDE1.....	64
FIG. 4.12	Exemplo de rede a ser editada.	65
FIG. 4.13	Remoção da ligação 10.	66
FIG. 4.14	Remoção do nó 4.....	67
FIG. 4.15	Seleção do ponto 3 para ser reposicionado.....	67
FIG. 4.16	Reposicionamento do ponto 3.	67
FIG. 4.17	Inserção de um novo ponto.....	67
FIG. 4.18	Seleção do ponto 5 para ser origem da nova ligação.	68
FIG. 4.19	Inserção da nova ligação, (5, 4).....	68
FIG. 4.20	Exemplo de grafo de SZWARCFITER (1986), utilizado como exemplo de rede com 13 pontos e 18 ligações.....	70
FIG. 4.21	Exemplo de SZWARCFITER (1986) traçado em coroa.	71
FIG. 4.22	Exemplo de SZWARCFITER (1986) traçado pelo Kamada..	72
FIG. 4.23	Exemplo de SZWARCFITER (1986) traçado em níveis gerado por busca em largura a partir do ponto 1, com ligações alternativas visíveis e posicionamento reto dos pontos.	73
FIG. 4.24	Exemplo de SZWARCFITER (1986) traçado em níveis gerado por busca em largura a partir do ponto 1, com ligações alternativas não visíveis e posicionamento reto dos pontos.	75
FIG. 4.25	Exemplo de SZWARCFITER (1986) traçado em níveis gerado por busca em largura a partir do ponto 1, com ligações alternativas não visíveis e posicionamento radial dos pontos.	76
FIG. 4.26	Tela “Visualizar...”	78
FIG. 4.27	Exemplo de arquivo texto para importação.....	80

FIG. 4.28	Tela “Exportar...”	83
FIG. 4.29	Arquivo de exportação do exemplo de SZWARCFITER (1986), sem as coordenadas dos pontos	83
FIG. 4.30	Arquivo de exportação do exemplo de SZWARCFITER (1986), com as coordenadas dos pontos	83
FIG. 4.32	Tela inicial do KitNet com cor de fundo cinza.	84
FIG. 4.33	Tela “Localizar Ponto...”	86
FIG. 4.34	Função IsOnEdge.	87
FIG. 4.35	Verifica se o ponto P seleciona a ligação (α, ω)	88
FIG. 7.1.1	Exemplo “A” – Traçado aleatório.	101
FIG. 7.1.2	Exemplo “A” – Traçado em coroa.	102
FIG. 7.1.3	Exemplo “A” – Traçado pelo Kamada.	102
FIG. 7.1.4	Exemplo “B” - Busca em largura sobre traçado aleatório, partindo do ponto 3 e com ligações alternativas visíveis. ...	103
FIG. 7.1.5	Exemplo “B” - Busca em largura sobre traçado aleatório, partindo do ponto 3 e com ligações alternativas não visíveis.	104
FIG. 7.1.6	Exemplo “B” - Busca em profundidade sobre traçado aleatório, partindo do ponto 3 e com ligações alternativas visíveis. ...	105
FIG. 7.1.7	Exemplo “B” - Busca em profundidade sobre traçado aleatório, partindo do ponto 3 e com ligações alternativas não visíveis.	106
FIG. 7.1.8	Exemplo “B” - Busca em largura sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas visíveis. ...	107
FIG. 7.1.9	Exemplo “B” - Busca em largura sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas não visíveis.	108

FIG. 7.1.10	Exemplo “B” - Busca em profundidade sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas visíveis.	109
FIG. 7.1.11	Exemplo “B” - Busca em profundidade sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas não visíveis.	110
FIG. 7.1.12	Exemplo “B” - Busca em largura sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas visíveis. ...	111
FIG. 7.1.13	Exemplo “B” - Busca em largura sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas não visíveis.	112
FIG. 7.1.14	Exemplo “B” - Busca em profundidade sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas visíveis.	113
FIG. 7.1.15	Exemplo “B” - Busca em profundidade sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas não visíveis.	114
FIG. 7.1.16	Exemplo “B” - Busca em largura sobre traçado em níveis com disposição reta, partindo do ponto 3 e com ligações alternativas visíveis.	115
FIG. 7.1.17	Exemplo “B” - Busca em largura sobre traçado em níveis com disposição reta, partindo do ponto 3 e com ligações alternativas não visíveis.	115
FIG. 7.1.18	Exemplo “B” - Busca em largura sobre traçado em níveis com disposição radial, partindo do ponto 3 e com ligações alternativas visíveis.	116
FIG. 7.1.19	Exemplo “B” - Busca em largura sobre traçado em níveis com disposição radial, partindo do ponto 3 e com ligações alternativas não visíveis.	116
FIG. 7.1.20	Exemplo “B” - Busca em profundidade sobre traçado em níveis com disposição radial, partindo do ponto 3 e com ligações alternativas visíveis.	117
FIG. 7.1.21	Exemplo “B” - Busca em profundidade sobre traçado em níveis com disposição radial, partindo do ponto 3 e com ligações alternativas não visíveis.	117

- FIG. 7.1.22 Exemplo “C” – Traçado aleatório, exceto pontos cujas coordenadas foram estabelecidas no arquivo original. Exibição sem grade e moldura..... 118
- FIG. 7.1.23 Exemplo “C” – Traçado níveis gerado por busca em largura a partir do ponto 33, com ligações alternativas visíveis e posicionamento reto dos pontos. Exibição com grade..... 119
- FIG. 7.1.24 Exemplo “C” – Traçado níveis gerado por busca em largura a partir do ponto 33, com ligações alternativas visíveis e posicionamento radial dos pontos. Exibição com grade. 119

RESUMO

Trabalhos na área de visualização e animação com o uso de ferramentas de informática começam a surgir no início dos anos 80, especificamente com trabalhos voltados para o ensino de informática. Com o passar dos anos são encontrados artigos, textos e ferramentas que envolvem visualização em engenharia de software e desenvolvimento de frameworks, análise, simulação e otimização de redes, tratamento e representação gráfica de grandes volumes de informações, de objetos abstratos e de relações entre objetos, projeto de web para sistemas distribuídos, validação de sistemas, ensino à distância, e ferramentas de auxílio à aprendizagem de algoritmos, grafos e linguagens de programação. O objeto desta dissertação é visualização de redes. Para tanto, encontra-se dividida em: organizar o material publicado sobre o tema; estudar a aplicabilidade dos aspectos relacionados à visualização em problemas reais modelados por redes; formalizar os procedimentos de visualização de redes e implementar o KitNet, um protótipo de ferramenta de visualização, ou melhor, um protótipo de software de visualização de redes.

ABSTRACT

Works in the visualization and animation areas with the aid of the computer science tools start in the beginning of years 80, specifically with works directed toward the computer science education. With passing of the years articles are found, texts and tools that involve visualization in software engineering and development of frameworks, analysis, simulation and otimização of nets, treatment and graphical representation of great volumes of information, abstract objects and relations between objects, project of web for distributed systems, validation of systems, long-distance education, and tools of aid to the learning of algorithms, graphs and programming languages. The object of this dissertation is visualization of nets. Therefore, one meets divided in: organization of the material published on the subject; study of the applicability of the aspects related to the visualization in real problems shaped by nets; standardization of the procedures of visualization of nets and implementation of the KitNet, a prototype of tool of visualization, or better, a prototype of software of visualization of nets.

1 INTRODUÇÃO

1.1 O QUE É VISUALIZAÇÃO

Visualização, segundo O Novo Dicionário Aurélio – Século XXI (FERREIRA, 1999), é definido como transformação de conceitos abstratos em imagens real ou mentalmente visíveis. Por sua vez, o radical "visualizar" significa formar ou conceber uma imagem visual mental de algo que não se tem ante os olhos no momento ou, ainda, tornar visível mediante manobra ou procedimento.

Contextualizando o termo para uso específico em sistemas de computação, PRICE et al. (1993), o definem como a capacidade ou processo de formação de uma imagem mental ou visão de algo que não está sendo visto no momento, podendo resultar da percepção de qualquer dos sentidos humanos. É interessante ressaltar que esta nova definição, apesar ser voltada para o contexto da informática, apenas reforça o conceito anterior, uma vez que não traz uma nova perspectiva ou especificidade. Assim sendo, a abordagem desta introdução é discutir tais conceitos, entre outros, e situá-los neste contexto mais específico.

Os termos “visualização” e “animação” representam conceitos diferentes. Entretanto, se confundem por serem inúmeras vezes utilizados em conjunto, por possuírem técnicas comuns a ambos ou por serem complementares. Além disso, são conceitos que se interceptam e se sobrepõem, sem, contudo, um conter o outro. Ou seja, a animação e a visualização não caracterizam uma relação de generalização ou de especialização entre si.

Na animação transfere-se para a imagem algum tipo de entendimento de fenômenos dinâmicos e, ainda, busca-se “dar vida a alguma coisa” (SORENSEN, 2001). Tal definição, sob nossa ótica, pode ser adotada quase totalmente para visualização. Todavia, devemos entender que, se desejado, podemos abandonar o dinamismo que a animação traz consigo, pois o objetivo final da visualização não é “dar vida”, mas permitir representações visuais de um objeto de estudo, um sistema ou um serviço.

A visualização é implementada através dos softwares de visualização (SV), que são ferramentas desenvolvidas com uso de técnicas de cinema, projeto gráfico e de desenho animado para apresentar estruturas de dados, programas ou algoritmos (PRICE et al., 1993). De acordo com DOMINGUE (1995), SV é basicamente a unificação de animação de algoritmos e visualização de programas, conceitos desenvolvidos nos anos 80.

Por sua vez, animação de algoritmos é a caracterização em alto nível de como os dados são manipulados durante a execução dos programas (DOMINGUE, 1995) ou, ainda, por BROWN (1988), animação de algoritmos mostra uma abstração em alto nível do código e dos dados de um programa. O que ambos os autores querem dizer com “caracterização em alto nível” e “abstração em alto nível” é apresentar uma forma de representação visual dos objetos manipulados.

Complementando, segundo DOMINGUE (1995), visualização de programas mostra o quanto a representação gráfica é fortemente acoplada com o código ou com os dados do programa, e apresenta com mais ou menos fidelidade como o código está sendo executado.

1.2 O QUE SÃO REDES DE SERVIÇO

Seja o grafo $G = (V, E)$, V o conjunto de vértices e E o conjunto de arestas de G . Segundo COULLARD et al. (1999), define-se rede, ou rede de serviço, como $N = (G, I)$, sendo I informações sobre os vértices e arestas de G . Redes são modelos matemáticos construídos para a representação de sistemas físicos ou abstratos (COULLARD et al., 1999).

A representação de serviços ou sistemas em redes apresenta uma visão da estrutura, da organização e do funcionamento dos mesmos, onde os pontos (vértices) da rede podem representar, segundo HE (1999), uma entidade “física” (uma cidade, um ponto de fornecimento de energia, um site, etc.) ou “não-física” (um IP, um domínio, um endereço de uma home page, etc.).

A ligação entre os pontos (a aresta) pode representar um link de comunicação, um cabeamento telefônico ou elétrico, etc., dependendo do sistema ou serviço representado, e, assim como os pontos, cada qual possui suas próprias características e informações. Estas, por sua vez, podem ser divididas em “permanentes”, como, por exemplo, a capacidade nominal máxima da ligação, e “temporárias”, como o fluxo na ligação em um determinado instante. Essas informações são valores agregados à entidade, não representando a entidade em si, que, como dito, pode nem mesmo existir fisicamente.

Uma rede é uma representação estática; entretanto, as entidades representadas (pontos e ligações) podem ter a necessidade de interagir com o passar do tempo, seguindo regras, protocolos, procedimentos e algoritmos em geral. Para tanto, basta que o sistema ou serviço representado possua natureza dinâmica, o que deverá provocar alterações no estado geral da rede e no estado das próprias entidades (memória local).

Redes de serviço podem ser redes de comunicação de dados, de telefonia, de energia elétrica, hidráulica, pluvial, uma malha rodoviária ou ferroviária, etc.. Levando em consideração tais exemplos, pode-se dizer que caracterizam-se por serem representadas por “locais ligados a outros locais”, uma “representação de caminhos”, e são estruturas que contêm informações e dados “escondidos”.

Particularizando o conceito de SV para software de visualização de redes (SVR), por HE (1999), define-se, em termos gerais, como sendo a apresentação de um conjunto de *vistas* das diferentes perspectivas do estado de uma rede, o que ressalta o caráter dinâmico dos estados de uma rede. Sob nosso ponto de vista, ressalta o caráter dinâmico das características dos elementos que a compõem.

O levantamento de trabalhos e aplicações relacionadas à visualização, realizado para a elaboração desta dissertação, apresentou um campo extremamente vasto, pois foram encontrados artigos, publicações e implementações que envolvem, entre outras, engenharia de software e desenvolvimento de frameworks (KRISHNAMOORTHY & SWAMINATHAN, 1989) (KAMADA & KAWAI, 1991) (HE, 1999), análise e simulação de rede (BECKER et al., 1990) (EICK & WILLS, 1993) (BECKER et al., 1995) (HE & EICK, 1998) (COULLARD et al., 1999) (HE, 1999), tratamento e representação gráfica de

grandes volumes de informações, de objetos abstratos e de relações entre objetos (BECKER et al., 1990) (KAMADA & KAWAI, 1991) (EICK & WILLS, 1993) (CARMO & CUNHA, 1998) (ALVES et al., 2000), projeto de web para sistemas distribuídos (DOMINGUE & MULHOLLAND, 1997b) (ALVES et al., 2000), validação de sistemas (principalmente sistemas baseado em conhecimento) (DOMINGUE, 1995), muitos trabalhos de ensino à distância (DOMINGUE & MULHOLLAND, 1997a) (DOMINGUE & MULHOLLAND, 1997b) (COULLARD et al., 1999) e ferramentas de auxílio à aprendizagem de algoritmos, grafos e linguagens de programação (MINCY et al., 1983) (BROWN & SEDGEWICK, 1984) (HIMSOLT, 1989) (HE, 1999) (DOMINGUE & MULHOLLAND, 1997a) (DOMINGUE & MULHOLLAND, 1997b) (MARKENZON & VERNET, 1997) (MARKENZON & VERNET, 1998) (COULLARD et al., 1999).

Diante dessa diversidade de aplicações nota-se que o conceito de redes de serviços deve ser ampliado, pois em alguns casos não apresenta uma representação de caminhos, normalmente mostrada pelos elementos genéricos da rede (vértices e arestas), mas uma relação de dependência entre as entidades representadas por esses elementos. Por exemplo, um grafo de alocação de recursos em um sistema operacional não modela uma rede, mas a dependência entre os vértices, que representam os processos e recursos do ambiente. Assim, por TANENBAUM & WOODHULL (2000), uma aresta partindo de um processo para um recurso representa a solicitação de uma instância do recurso e, o inverso, representa a atribuição da instância ao processo.

Como exemplo de redes de serviços passíveis de representação podemos citar: redes hidráulica, de energia elétrica, telefônica ou de comunicação de dados, malhas rodoviária ou ferroviária, rotas marítimas ou aéreas, planta de uma cidade ou bairro, malhas pluvial ou fluvial, grafo de alocação de recursos, diagrama de fluxo, fluxo em redes, etc..

1.3 O QUE SÃO SOFTWARES DE VISUALIZAÇÃO

Segundo DOMINGUE et al. (1992), a visualização ajuda engenheiros de software a monitorar e analisar programas através de ambientes de depuração, que são utilizados para acompanhar o código fonte e examinar o comportamento do programa em pontos de parada arbitrários e o conteúdo das estruturas de dados. Entretanto, ressaltam que tais aplicações têm o entendimento limitado aos especialistas da área.

Ainda de acordo com DOMINGUE et al. (1992), os trabalhos de BAECKER & SHERMAN (1981) e BROWN (1988) iniciam uma nova fase, mostrando como algoritmos podem ser animados como “cartoons”, através da representação visual abstrata dos objetos manipulados, buscando ampliar o entendimento para além de especialistas, isto é, *animação de algoritmos* (BROWN, 1988) (seção 1.1).

Muitas ferramentas de visualização foram desenvolvidas, principalmente a partir de meados da década de 1980, dando origem aos “programas animadores”, “programas de visualização”, “animação de algoritmos”, “linguagens de visualização”, etc., quando, então, naturalmente surgiu a demanda pela necessidade de classificação e diferenciação dos mesmos. O termo *software de visualização*, contudo, somente passou a ser utilizado fazendo referência às ferramentas já existentes a partir de PRICE et al. (1993), que, no mesmo trabalho, propuseram uma taxonomia bem aceita atualmente.

Com relação a classificação, organização de termos e taxonomia, de acordo com DOMINGUE et al. (1992), já em 1986, MYERS (1986) criou a primeira taxonomia que diferenciava SV, programação visual e programação por exemplos, na qual usou duas dimensões: a) estática e dinâmica; b) código e dados.

A primeira dimensão baseia-se no estilo da implementação, onde apresentações estáticas mostram um ou mais gráficos sem movimento, que representam o estado do programa em um ponto particular no tempo, enquanto apresentações animadas (dinâmicas), mostram imagens que se modificam durante a execução do programa.

A segunda dimensão refere-se ao tipo de dado a ser visualizado, que pode ser o código fonte do programa em execução ou as estruturas de dados que o programa manipula.

De forma abrangente, com relação às propostas de classificação e organização, DOMINGUE et al. (1992) criticam as taxonomias que apresentam poucas dimensões, pois essas ignoram o fato de existirem muitos estilos de implementação e de interação, assim como o fato de existirem diferentes arquiteturas de sistemas e as suas formas de utilização.

PRICE et al. (1993) propõem seis categorias para classificação de SV: *escopo, conteúdo, forma, método, interação e efeito*. Sendo que cada categoria possui entre três e sete características, perfazendo um total de trinta dimensões para descrever um sistema. Entretanto, DOMINGUE et al. (1992) também questionam essa classificação, que já havia sido apresentada, em 1992, na 25th Hawaii International Conference on System Sciences, e alerta que essa pragmática classificação de sistemas fornece meios para comparar a funcionalidade e o desempenho de uma vasta gama de SV, mas não provê uma linguagem ou framework para a implementação de novos sistemas.

Por sua vez, segundo DOMINGUE et al. (1992), EISENSTADT et al. (1990), sob outra ótica, descrevem nove dimensões qualitativas, as quais podem ser a base de uma linguagem descritiva de SV, mas se aplicam somente para descrever os atributos de sistemas, não sendo direcionado para construção dos mesmos.

1.4 PROPOSTA DE TRABALHO

O objeto de nosso estudo é visualização de redes, considerando que, na presente dissertação, visualização objetiva facilitar o entendimento dos dados e informações que trafegam pela rede, ou seja, os objetos que são manipulados e alterados pelo serviço representado, e não a rede em si.

Para tanto, será organizado o material publicado sobre o tema, estudar a aplicabilidade dos aspectos relacionados à visualização em problemas reais

modelados por redes, formalizar os procedimentos de visualização de redes e implementar um protótipo de ferramenta de visualização, isto é, um protótipo de software de visualização de redes (SVR).

O presente capítulo iniciou-se com a apresentação dos fundamentos necessários, tais como, visualização, redes de serviço, e software de visualização, e, ainda, foram situados dentro do contexto de sistemas de computação.

No segundo capítulo abordaremos os trabalhos existentes na literatura, demonstrando a abrangência do assunto, quando alguns trabalhos serão investigados mais profundamente, e, como consequência, relataremos as conclusões do levantamento.

No terceiro capítulo apresentaremos o KineGraph, justificando seu uso e, em seguida, faremos uma avaliação crítica do mesmo. Prosseguindo, o quarto capítulo apresentará a proposta, descreverá e relatará a implementação do protótipo, o KitNet, e, também será feita uma avaliação crítica do mesmo. Por fim, o quinto capítulo apresentará as conclusões e sugestões de trabalhos futuros.

O protótipo desenvolvido iniciou-se com o porte do código fonte da biblioteca de rotinas básicas do software KineGraph, originalmente desenvolvido na linguagem Pascal. Em seguida a implementação partiu para os procedimentos de “criação” da rede a ser visualizada, através de geração automática e aleatória de conjuntos de dados, e os procedimentos de “criação interativa” da rede e de edição da rede, indiferentemente se esta for criada de forma interativa ou automática.

Também considerou-se importante disponibilizar alguns tipos de traçados para o desenho da rede. Assim, o protótipo permite os traçados aleatório, de coroa, hierárquico e, ainda, o traçado Euclideano, que visa harmonizar o desenho em função do tamanho das ligações, ou melhor, em função do comprimento dos segmentos de retas que representam as arestas. Por último foram desenvolvidas as interfaces entre os dados manipulados pelo KitNet, pelo KineGraph e um padrão em arquivo texto.

2 TRABALHOS EXISTENTES NA LITERATURA

2.1 INTRODUÇÃO E REVISÃO BIBLIOGRÁFICA

Já foram apresentadas as diversas áreas nas quais os conceitos e técnicas de visualização são utilizadas em trabalhos, artigos e softwares, onde mostrou-se o crescente interesse na área, até mesmo ao verificarmos a proporção de trabalhos recentes em comparação com o percentual de trabalhos mais antigos. Assim, a seguir, será feito um breve relato de importantes trabalhos organizado cronologicamente, visando destacar o crescimento do aporte tecnológico envolvido em cada trabalho, chegando ao uso de arquitetura de sistemas distribuídos via web, nos últimos anos.

Um dos primeiros trabalhos relevantes, desenvolvido por MINCY et al. (1983), mostra e justifica como a visualização pode ajudar no ensino de algoritmos. Além disso, apresenta um protótipo de ferramenta de visualização e exemplifica como esse SV pode ser usado como instrutor. Posteriormente veremos mais detalhes sobre esse trabalho.

Apesar de ser um trabalho independente, e sem citação por parte de outros autores, HIMSOLT (1989) desenvolveu um editor interativo para grafos que ilustra bem a dependência tecnológica das ferramentas de visualização. Este trabalho também será apresentado com mais detalhes, na seção 2.2, e será possível perceber o quanto um software abrangente, em um determinado contexto de época, pode tornar-se tão simples que poucos procedimentos de outros softwares cobrem toda a abrangência do primeiro, em outro contexto.

O primeiro grande SV interativo, segundo PRICE et al. (1993), foi o Brown University Algorithm Simulator and Animator (BALSA) apresentado BROWN & SEDGEWICK (1984), sendo que suas versões posteriores foram utilizadas por Brown em sua premiada tese (BROWM, 1988). Sabe-se da importância desse autores para animação e visualização, visto que essas áreas somente começaram a tomar forma, tal qual se apresentam hoje, a partir de seus

trabalhos. Em especial, no citado artigo os autores mostram como o Balsa possibilita apresentar, muitas vezes com uma representação em alto nível de abstração, as propriedades de um programa em execução, através de algoritmos de animação associados às estruturas de dados.

No enfoque de engenharia de software, KRISHNAMOORTHY & SWAMINATHAN (1989) propõem um conjunto de primitivas para o desenvolvimento de softwares de visualização e animação, justificando que seus experimentos mostram que, com o uso do conjunto proposto, diminui-se o tempo de desenvolvimento. Neste artigo, o conjunto de primitivas visa permitir de forma simplificada a manipulação e a representação gráfica de objetos e, ainda, a interação desses objetos com os eventos de entrada.

Em visualização de redes, foi visto o trabalho de BECKER et al. (1990), o qual, de início, afirma que informações sobre redes, ou melhor, sobre os nós e as ligações, estão associadas a dados estatísticos, e são originados a partir do levantamento do fluxo na rede, acesso aos nós, etc.. Porém, os autores defendem que, caso fosse utilizada uma representação expressiva, por exemplo, cores com significados definidos para os nós e as ligações, e, além disso, se a rede fosse disposta visualmente sobre seu contexto, por exemplo, uma malha rodoviária visualizada sobre um mapa, facilitaria a leitura e o entendimento da rede.

O problema nesta proposta é que possivelmente tem-se “um pequeno número de linhas e símbolos contrapondo-se a um grande volume de dados, o que pode tornar-se ilegível” (BECKER et al., 1990). Todavia, os autores apresentam a solução através de duas técnicas gráficas dinâmicas para visualização de dados de redes. A primeira tratando os dados das ligações e a outra os dados dos nós, ambas com o objetivo de “limpar” a visualização da rede.

Novamente sob o enfoque de engenharia de software, KAMADA & KAWAI (1991) apresentam um framework para a representação visual de objetos abstratos e suas relações, e uma interface de visualização baseada nessa estrutura. Os autores defendem o uso da visualização, aqui denominada *representação pictorial*, como ferramenta que facilita a representação da informação - “... a picture is worth a thousands words...” (KAMADA & KAWAI,

1991) - visto que a visualização amplia significativamente a capacidade de entendimento de complexas relações entre estruturas.

O framework proposto permite traduzir objetos abstratos e suas relações, normalmente representados de forma textual, para representações pictoriais, através do mapeamento dos objetos abstratos em objetos gráficos e as relações entre os mesmos são definidas por regras de mapeamento. Ressaltam que um SV requer grande aporte computacional para permitir visualizar vários tipos de informações e uma grande variedade de representações gráficas, porém defendem que a proposta vem contornar tal dificuldade.

DOMINGUE et al. (1992) discutem os SV desenvolvidos nos últimos anos que possuem como objetivo melhorar a interface entre os engenheiros de software e seus programas. Também discutem três classificações taxonômicas de softwares de visualização e, ainda, apresentam a ferramenta Viz, desenvolvida pelos autores.

O Viz considera programas em execução como uma série de eventos históricos acontecendo para os usuários ou perpetrados por esses. Para possibilitar a visualização dos eventos existem mecanismos de tratamento, onde os quatro principais, a partir de um ponto de vista macro, são: os históricos, registros dos principais eventos que ocorrem durante o tempo de execução do programa; as vistas, o estilo sob o qual um conjunto particular de usuários, estados ou eventos são apresentados; os mapeamentos, as codificações utilizadas por um usuário para mostrar a mudança de estado, na forma de textos ou diagramas, que podem ocorrer em uma vista ao longo do tempo, utilizando algum tipo de linguagem gráfica, tipográfica ou sonora; e, por fim, os navegadores, que são ferramentas ou técnicas que possibilitam ao usuário mover-se por entre as vistas, alterar a escala do desenho, comprimir ou expandir objetos e movimentar-se pelo histórico de eventos.

Segundo seus autores, o principal objetivo deste trabalho, aliás, concordando com o título deste artigo, é criar um framework para descrição de sistemas existentes e para a implementação de novos. Portanto, o projeto do Viz partiu do entendimento e explanação de diversas notações e metodologias existentes em ambientes de SV, tais como, o BALSÁ e o TANGO, que serão apresentado

posteriormente, até estabelecer mecanismos que permitissem o desenvolvimento de uma ferramenta de descrição.

EICK & WILLS (1993) apresentam um excelente artigo que trata de visualização de redes e de representação de grandes volumes de informações, aliás dando prosseguimento a BECKER et al. (1990), onde atuaram como co-autores.

Neste novo trabalho, os autores exploram a visualização de redes de sistemas de computação, utilizando não só cores com significado conhecido para representar informações relativas aos nós e ligações, visto que são associados às ligações “força” e “peso” e aos nós uma determinada hierarquia, mas, também, descrevem e utilizam um método de posicionamento dos nós no plano que confere significado relativo ao posicionamento.

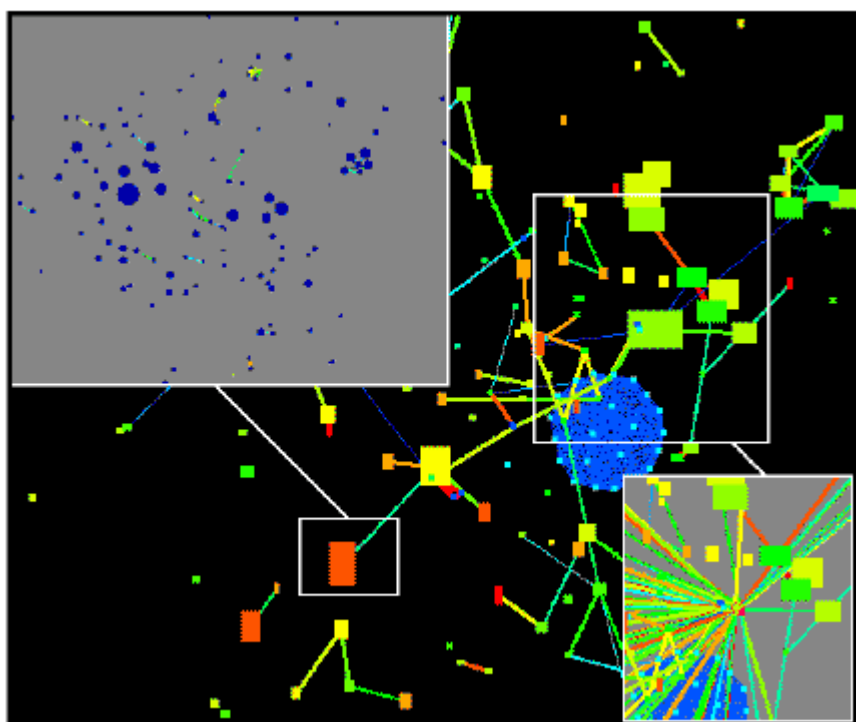


FIG. 2.1: Expansão e contração de nós dentro da hierarquia (EICK & WILLS, 1993).

O posicionamento não segue uma localização pré-definida, por exemplo, geográfica, e nem se preocupa em torná-la harmoniosa, mas considera o peso das ligações entre os nós, fazendo com que nós fortemente ligados fique próximos e, assim, conferindo um significado relativo ao posicionamento. Os autores ainda observam que como um bom efeito colateral, consegue-se uma certa harmonia, pois dessa forma reduz-se a possível desordem na visualização.

O tratamento de grandes volumes de informações é feito através da interpretação da hierarquia. A hierarquia representa o agrupamento de objetos relacionados, por exemplo, se representarmos graficamente o desenvolvimento de um software como uma rede, ou simplesmente um diagrama de relacionamento entre partes do código fonte, temos os arquivos fontes, que podem ser estruturados em módulos, e estes, por sua vez, formam subsistemas. Assim, pode-se reduzir o volume de informações a serem representadas se for feita a contração (“zoom-out”) dos arquivos fontes em seus respectivos módulos e, neste caso, apresentar-se-ia a visualização dos módulos, e não dos arquivos fontes que o compõem, e, como consequência, será visto a interação do módulo com os demais objetos do sistema, e não a interação de cada arquivo fonte com os demais objetos (FIG. 2.1).

Em um segundo momento, caso seja conveniente, a visualização poderia ser reduzida a apresentação dos subsistemas, o que reduziria ainda mais os objetos representados e a interação entre os mesmos.

Um dos trabalhos que também marca o estudo da visualização é o artigo de PRICE et al. (1993), pois objetiva apresentar uma detalhada taxonomia de SV, após, aproximadamente, uma década de trabalhos publicados e desenvolvidos, e analisa doze ferramentas de visualização.

Os autores justificam a necessidade de uma taxonomia bem formulada para estabelecer a terminologia e uma linguagem comum e facilitar a troca de idéias e novas descobertas. Além disso, vão mais além quando afirmam que o uso de uma taxonomia pode discernir o que aparentemente é uma nova descoberta ou somente um refinamento ou variação de algo que já existe.

Esse trabalho foi importante devido às definições que sugere para muitos termos utilizados, ou em outros casos uma nova interpretação para termos já definidos. Por exemplo, os autores não diferem visualização de algoritmos e animação de algoritmos, e definem ambos de forma superficial, como sendo a:

“visualização de uma descrição em alto nível de uma ‘peça’ de software”.

Claramente essa definição deriva, de forma simplificada, da apresentada por BROWN (1988) (seção 1.1).

Algumas das ferramentas descritas por PRICE et al. (1993), são: o Brown University Algorithm Simulator and Animator (BALSA); o Zeus, uma das evoluções do BALSA; o TANGO, também desenvolvido na Brown University; o ANIM, desenvolvido na AT & T Bell Laboratories; o Pascal Genie (Amethyst), desenvolvido em Carnegie-Mellon juntamente com um protótipo da Xerox PARC; o University of Washington Program Illustrator (UWPI); e o Transparent Prolog Machine (TPM), da U. K. Open University.

Também são apresentadas ferramentas de apoio ao desenvolvimento, tais como ambientes de desenvolvimento integrados (com editores, debugadores, janelas de inspeção de valores de variáveis, etc.), citam ainda o SEE, que, na verdade, é um formatador de código fonte, ou seja, lê um programa fonte e formata o arquivo fonte para melhorar a visualização, através de indentação para destacar escopo de blocos, inserindo linhas para facilitar a leitura. Finalmente destacam como o primeiro grande trabalho de visualização, apesar de não ser um software, o “Sorting Out Sorting” (SOS). O SOS é um vídeo educacional produzido na Universidade de Toronto que utiliza animação em computação gráfica para explicar como nove diferentes algoritmos de ordenação (inserção linear, método da bolha, inserção binária, seleção, “shakersort”, árvore de seleção, “shellsort”, “quicksort” e “heapsort”) manipulam os dados.

Novamente BECKER et al. (1995) desenvolvem um trabalho relacionado com visualização de redes e apresentam o SeeNet, um SVR que trata e valida a abordagem proposta para visualização de dados associados a redes de grandes dimensões. Enfatizam a abundância de informações possíveis de serem retiradas de uma rede e apontam a visualização como uma grande estratégia para o entendimento da mesma, ressaltando, assim como em nossa abordagem, a importância de visualizar a rede associada a seus valores e não visualizá-la por si só.

O SeeNet permite três diferentes vistas de uma rede, duas relacionadas à disposição geográfica da rede e a outra uma disposição matricial, que podem ser parametrizadas, de forma interativa, e assim permitir análise e descobertas sobre o comportamento da rede.

Estando ciente de que a visualização é uma área dependente da tecnologia de recursos computacionais, de plataformas de hardware, de linguagens e

técnicas de programação, é fácil supor que muitos trabalhos tendem a tornar-se obsoletos em um curto espaço de tempo. Além disso, é natural que novos recursos computacionais permitam a expansão ou ampliação dos trabalhos.

Assim, HE & EICK (1998) partem de trabalhos anteriores, onde os autores destacam entre outros BECKER et al. (1995), e apresentam um excelente SVR, aliás, como definem, uma interface visual e interativa para redes, cujo requinte de apresentação gráfica é um diferencial, permitindo inclusive a visualização em 3D (FIG. 2.2), quando obviamente deve-se optar pela representação de menos informações.

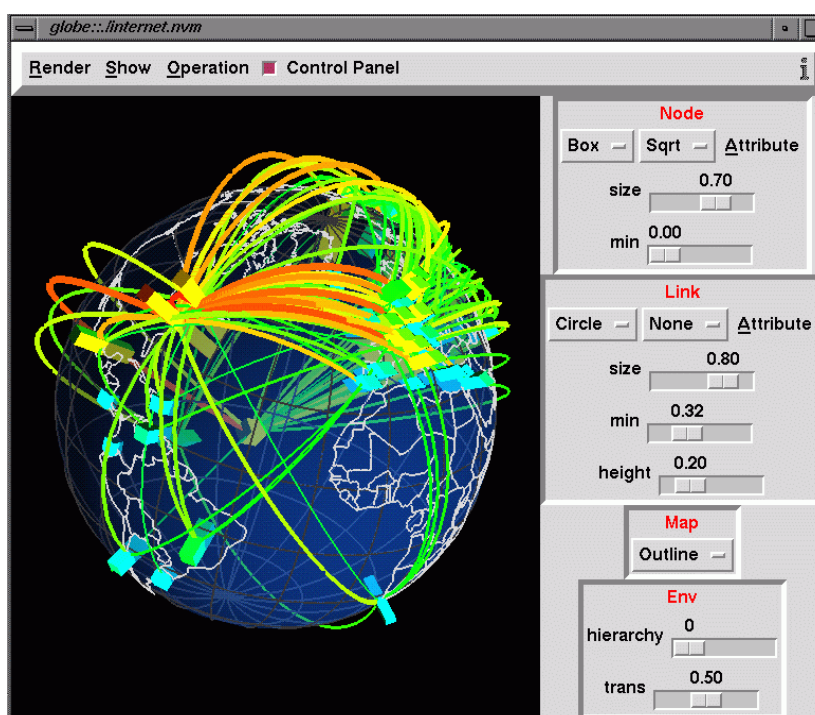


FIG. 2.2: Exemplo de uma vista possível em uma rede global hipotética (HE & EICK, 1998).

Os autores tiveram a correta preocupação como desenvolvimento multiplataforma e então desenvolveram o núcleo da ferramenta em C e C++ e a interface gráfica do usuário (GUI) em Tcl/Tk, o que permite a compilação da ferramenta em ambientes Unix e Windows.

O SV é um sistema orientado a objeto para a visualização interativa de redes complexas, hierárquicas e que apresentam variação temporal de seus parâmetros, segundo os autores. Entretanto, apesar da expressão “variação temporal”, entre as sugestões de trabalhos futuros encontra-se a proposta de um framework que suporte conexões com bases de dados em tempo real para que o

sistema assuma características dinâmicas e, assim, por exemplo, possa ler e representar os dados de um simulador de rede. Assim, podemos supor que a citada expressão não indica interação dinâmica do ponto de vista das informações, como poderíamos imaginar em um primeiro momento.

Esse artigo também é muito rico no que tange a definições visto que propõe uma taxonomia para as operações básicas de visualização, que posteriormente será tratada em detalhes.

MARKENZON & VERNET (1998) citam alguns problemas comuns de serem enfrentados quando se trabalha com algoritmos em grafos e propõem uma solução, através do KineGraph, uma ferramenta para implementação de tais algoritmos. O KineGraph, assim como o trabalho em questão, serão abordados com mais critérios no capítulo 3, devido a importância desta ferramenta para o presente texto.

Um dos grandes problemas de visualização é a representação de grandes volumes de informações, ainda mais se for considerado o crescimento quantitativo das informações disponíveis, bem como a crescente variação das formas que se apresentam. Sob este problema escondem-se, na verdade, dois problemas: a filtragem, ou seleção, da informação a ser apresentada, e a representação da informação. Muitos trabalhos aqui relacionados, por exemplo, BECKER et al. (1990), EICK & WILLS (1993), CARMO & CUNHA (1998) e ALVES et al. (2000), abordam estes pontos com maior ou menor ênfase, entretanto, o artigo de CARMO & CUNHA (1998) trata especificamente o tema e, portanto, será detalhado na seção 2.2. Este mesmo artigo faz uma excelente descrição de modelos de visualização, partindo de técnicas de filtragem, seleção e de representação.

Como mencionamos anteriormente, na análise de HE & EICK (1998), os autores sugeriram um trabalho futuro em SVR, com maior suporte à interatividade e voltado a tratar o dinamismo das modificações do estado de uma rede. Assim, HE (1999) continua seu trabalho desenvolvendo uma interface visual e interativa, desenvolvido em Java, para um simulador de rede proprietário da Lucent Technologies, suportando controle de tempo real e monitoramento remoto do simulador, via browser.

O projeto deste software buscou atender a quesitos como portabilidade e suporte a programação, em rede em ambiente orientado a objetos. Além disso, os autores desenvolveram e discutiram um framework para desenvolvimento de interfaces de rede. Este trabalho será apresentado com mais destaque na próxima seção.

Outro SVR que também merece destaque é o Graphical Implementation Development Environment for Networks, GIDEN, desenvolvido por COULLARD et al. (1999), pesquisadores da Universidade de Northwestern e do Sistema de Pesquisa de Ann Arbor, em Michigan.

Segundo os autores, o GIDEN é um ambiente de software interativo projetado para facilitar a visualização de problemas, soluções e algoritmos de otimização de redes, e, para tanto, oferece uma interface gráfica para construção e edição de redes, facilidades para animação de algoritmos, permitindo a implementação de algoritmos animados, e uma biblioteca de estrutura de dados relacionada com redes.

Além disso, também oferece exemplos já implementados, em algoritmos animados, dos problema da árvore de espalhamento mínimo (“minimum spanning tree problem”), do menor caminho (“shortest path problem”), do fluxo máximo (“maximum flow problem”) e do fluxo de custo mínimo (“minimum-cost flow problem”).

Uma de suas boas característica é o desenvolvimento para diferentes plataformas hospedeiras, pois o GIDEN foi desenvolvido para execução em ambientes Java, e a atual versão encontra-se disponível para configurações que suportem os sistemas operacionais MS Windows 95/98/NT, Solaris (sobre arquitetura SPARC) e outras implementações de Unix, OS/2 Warp, e, ainda, outras plataformas não relacionadas, porém que disponibilizem ambiente Java.

Outras características que valorizam este SVR, são a sua ótima apresentação visual e interface amigável, como pode ser comprovado nas FIG. 2.3 e 2.4, e, além disso, tendo sido desenvolvido com base em conceitos linguagens atuais, com poucas restrições quanto a plataforma de execução, visa ser depreciado o mínimo possível com o passar dos anos, buscando evitar que as mudanças tecnológicas influenciem negativamente o seu uso ou vida útil.

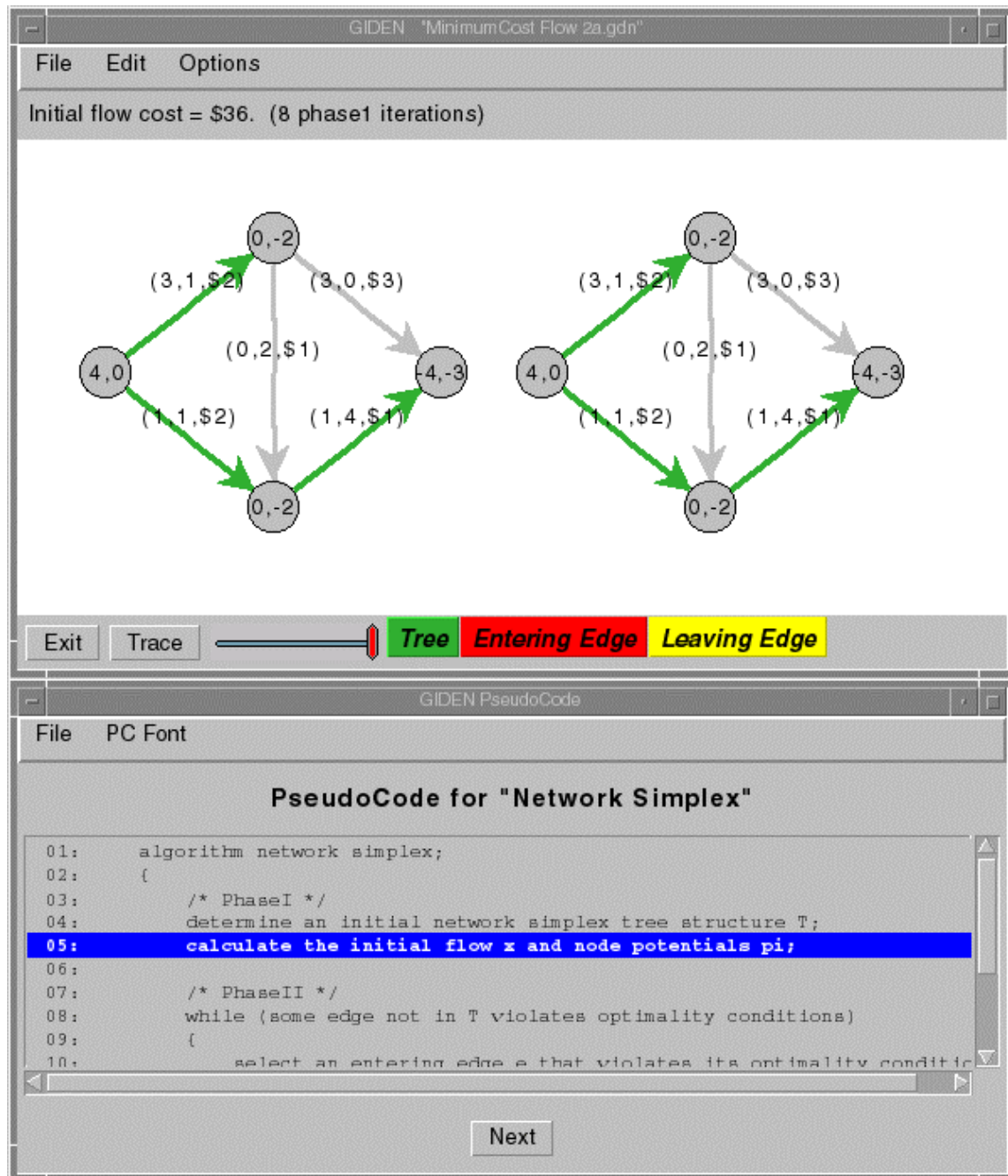


FIG. 2.3: Exemplo de visualização de uma rede com a execução animada de um algoritmo (COULLARD et al., 1999).

Com relação aos objetivos deste SV, os autores explicam que, como definido na seção 1.2, redes são modelos matemáticos construídos para a representação de sistemas físicos ou abstratos. Assim, otimização de redes significa construir uma rede para modelar um determinado sistema, indicando o problema nos termos do modelo da rede, e, então, resolver o problema detectado a partir de

algoritmos exatos ou heurísticos. Uma vez obtida a solução para o problema do modelo, é possível planejar uma solução real para o problema original.



FIG. 2.4: Menu de opções de edição da rede (COULLARD et al., 1999).

No XIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagem, realizado em 2000, foi apresentado um interessante trabalho que trata da visualização em ambientes web. Na verdade, os autores, ALVES et al. (2000), apresentaram o SV VisWeb, uma ferramenta de visualização totalmente voltada para aplicações web, todavia, o SV foi fruto da discussão dos dois grandes problemas da visualização, com já foi citado ao longo do texto, a representação de grandes volumes de informações e a plataforma operacional do software.

Com relação à representação de informações, os autores apresentam uma proposta que basicamente constitui-se de um processo dividido em etapas distintas ("pipeline") para geração da imagem que será visualizada, sendo composto por uma série de procedimentos com fins específicos: filtragem, mapeamento e "rendering"¹, para, então, nesta ordem, produzirem a representação visual esperada. Esse processo é similar ao apresentado por CARMO & CUNHA (1998) e, assim, será devidamente explicado, bem como os procedimentos, na próxima seção, quando apresentaremos tal artigo.

Quanto a plataforma operacional, o VisWeb foi desenvolvido com o uso de applets Java, sendo inerentes a tal ambiente, a característica negativa de, em geral, apresentar uma execução mais lenta, se comparado com executáveis

¹ Criação de superfície ou plano no processo de representação, ou seja, a geração da imagem propriamente dita.

gerados a partir de linguagens compiladas, e como característica positiva a portabilidade, que traz grandes vantagens ao apresentar-se praticamente independente de plataforma operacional.

Seus recursos de apresentação em três dimensões foram implementados a partir da biblioteca Java Abstract Windowing Toolkit, uma API desenvolvido e disponibilizada pela Sun Microsystems. Apesar de problemas inerentes à plataforma, a escolha foi muito bem feita, pois mostra-se funcional, independente, com utilização relativamente fácil e de baixo custo.

Além dos pontos levantados pelo artigo em questão, nota-se que seu conteúdo serve para exemplificar como o uso de tecnologias atuais e técnicas recentes, por exemplo, de desenvolvimento para web, podem ser empregados para a implementação recursos de visualização interativa, através de um sistema com arquitetura cliente/servidor.

2.2 TRABALHOS DE MAIOR DESTAQUE

Entre os trabalhos analisados, foram escolhidos alguns que mereceram destaque determinado por algum critério ou quesito acadêmico, ou pelo trabalho vir a representar um diferencial para a época em que foi desenvolvido ou publicado. Em linhas gerais, o critério foi determinado a partir da observação da concentração de trabalhos por área de aplicação ao longo da história, isto é, ao longo da evolução dos conceitos e técnicas de visualização ou animação.

Assim, os primeiros trabalhos em ordem cronológica destacaram a importância da visualização como ferramenta de auxílio à aprendizagem e entendimento da execução de algoritmos, e como destaque selecionamos o trabalho de MINCY et al. (1983), o qual também surpreende a atualidade de suas questões e argumentações em um trabalho datado de 1983.

O trabalho de HIMSOLT (1989) também mereceu destaque pelo desenvolvimento de uma ferramenta voltada ao ensino, porém específica para grafos. Além disso, exemplifica o quanto curta é a vida útil para SV, pois percebe-

se que a dimensão do trabalho perdeu-se rapidamente com a evolução tecnológica.

A escolha de CARMO & CUNHA (1998), além de também ser de cunho cronológico, foi determinada por tratar do grande problema da visualização, levantado por diversos trabalhos a que tivemos acesso: a representação de grandes volumes de informações. O trabalho também se mostra bastante interessante a medida de que os autores, além de proporem uma abordagem própria à questão, relacionaram diversas técnicas para o tratamento do problema que foram encontradas em outras publicações.

O último trabalho para o qual foi dado destaque trata especificamente do tema desta dissertação, visualização de redes. HE (1999), apresenta um “front-end” para um produto específico, um analisador de rede de comunicação de dados, o que pode não ser um ponto positivo. Entretanto, a implementação deste gerou um framework para a construção de ferramentas de visualização e prossegue até questões mais fundamentais, quando é proposta uma relação dos eventos e procedimentos que SV devem conter, a qual o autor apresenta como uma taxonomia.

O trabalho de MINCY et al. (1983) apresenta vantagens da visualização de algoritmos e procedimentos no processo de ensino e aprendizado da ciência da computação, e deve ser destacado que, apesar de passadas duas décadas, a argumentação, a justificativa e o enfoque do trabalho ainda são atuais, onde se vê que vários outros trabalhos seguiram a mesma linha, muitas vezes até os mesmos exemplos, alterando-se somente as soluções e implementações, não o conceito básico.

MINCY et al. (1983) partem do princípio de que o entendimento de algoritmos, processos e métodos é o centro das disciplinas a serem estudadas na ciência da computação, e formulam a hipótese inicial de que a tradicional abordagem de leitura dos algoritmos não é um mecanismo efetivo e conveniente, tornando-se menos efetiva à medida que a complexidade do algoritmo aumenta.

A dificuldade básica na apresentação de um algoritmo é sua representação clara na dimensão do tempo. A execução de um algoritmo provoca modificações no valores ou estados das entidades que manipula no decorrer do tempo, e a "sobreposição" desses valores atrapalha o entendimento do algoritmo. Como

distinguir uma etapa de outra quando se está tomando notas e como se representa o fator tempo? A chave do entendimento é a visualização.

A partir das justificativas expostas são apresentados dois protótipos desenvolvidos para auxiliar no ensino da disciplina de estrutura de dados: um para ensinar busca binária (com recursão) e outro para acompanhar a busca e identificação de padrão entre cadeias de caracteres. Entretanto características técnicas e funcionais não são abordadas.

Outra trabalho que mereceu destaque foi desenvolvido por HIMSOLT (1989). Este trabalho configura bem a questão de disponibilização de recursos em função de um determinado contexto que envolve, entre outros, os recursos tecnológicos, plataformas e técnicas de desenvolvimento de software. Percebe-se um trabalho bastante arrojado para a época, entretanto, hoje, mostra-se simples e com poucos recursos, sendo a totalidade de suas funções e características, praticamente igual a poucos procedimentos de um SV atual.

HIMSOLT (1989) apresenta um editor de grafo interativo, chamado Graph^{Ed}, para desenho e manipulação de grafos e de “gramáticas de grafos”, que permite o desenho de grafos, diagramas de entidade-relacionamento, redes de petri, grafos de fluxo e outros diagramas utilizados em projetos. O editor transforma uma lista de adjacência (a representação interna do grafo) para a forma visual e permite grafos direcionados ou não, arestas múltiplas e definição de rótulos.

Os objetos primitivos de grafos são nós, arestas e rótulos, sendo que cada um desses objetos é manipulado de forma flexível. Por exemplo, nós podem ser de formatos e estilos arbitrários, utilizando um construtor de estilo e ícones para o formato e, ainda, o usuário pode criar novos tipos de nós através de novos ícones, com tamanhos variados.

Rótulos também são manipulados de várias formas, sendo apresentados em caixas de tamanho definido pelo usuário, e são automaticamente anexados aos nós e arestas. No caso dos nós, o tamanho das caixas determina o tamanho dos nós e é possível colocar o rótulo no centro do nó ou em um dos quatro cantos.

Arestas são desenhadas como linhas retas ou pela junção de linhas, são anexadas aos nós em uma de cinco possíveis formas e a seta de uma aresta de um grafo direcionado também poder ser alterada em formato e tamanho.

Os procedimentos previstos na manipulação dos grafos são inserção, deleção, cópia e troca de nós e arestas, incluindo seus rótulos, além de alteração de tamanho, de estilo e de posicionamento. Nós podem ser movidos, redimensionados ou removidos sozinhos ou em conjunto, assim como os rótulos de um conjunto de nós podem ser alterados simultaneamente.

O programa pode comunicar-se com outros através de arquivos com formatos baseado em listas de adjacência, assim outros programas não precisam conhecer nada sobre o Graph^{Ed}, apenas sobre listas de adjacência. Pelo contrário, o Graph^{Ed} aceita grafos com apenas as posições dos nós especificadas e, então, passa a inferir pelas demais informações necessárias.

Uma das claras desvantagens da ferramenta apresentada é a de não permitir a valoração dos objetos que manipula, por exemplo, uma aresta não pode receber a indicação do custo na transição entre dois nós, visto que o rótulo é utilizado somente para identificação da aresta.

O artigo de CARMO & CUNHA (1998) merece destaque pela importância e natureza do problema que aborda, bem como pela solução proposta. É abordado o tratamento de grandes volumes de informações, sendo proposto um mecanismo de seleção conjugado à representação da informação, ou melhor, à escala de representação e à densidade da informação.

Com mais detalhes, CARMO & CUNHA (1998) apresentam um artigo onde a tônica é o método proposto para tratar e visualizar grandes volumes de informações, o que requer a existência de meios para reduzir a quantidade de informação a visualizar, quer eliminando informação, por filtragem, ou simplificando a forma de representá-la, através de escolha de representação.

Os autores propõem um modelo no qual relaciona-se funções de grau de interesse aos objetos a serem representados, sendo essas funções as responsáveis pelas diferentes representações adaptadas à escala usada. Ou seja, automatiza-se a medida do grau de interesse, tornando-se desnecessárias observações individuais para os objetos a serem visualizados. Assim, com a medida do grau de interesse automatizada, não será necessário criar novas representações ou recorrer a atributos para indicar o grau de interesse de cada elemento de informação. Por exemplo, inclusão de mecanismos de ampliação ou distorção permite aproveitar a redução de ocupação nas áreas onde há elementos

com menor interesse, aumentando a área destinada aos elementos com maior interesse.

Sabe-se que para visualizar grandes volumes de informação é necessário ter meios para reduzir o volume de informação a representar, seja através de filtros, da identificação de diferentes níveis de interesse, da redução do número de variáveis a representar, por eliminação ou agrupamento de variáveis, ou utilizando-se de diferentes simbologias e convenções com nível de detalhe, que variam com a escala ou com a importância do objeto.

Entre os problemas a serem tratados, a possibilidade de existir diferentes simbologias para o mesmo objeto, por exemplo em função de mudança de escala ou pelo agrupamento de objetos em um único bloco, obriga a existência de mecanismos para a seleção da representação mais adequada. Além disso, quando se trata da representação de objetos com dimensões e localizações precisas, a escolha da representação está condicionada essencialmente pela escala a usar. Assim, ainda por CARMO & CUNHA (1998), podem surgir problemas na visualização devido ao grande volume de informação, sendo necessário reduzir a informação a visualizar, ou podendo provocar uma representação ininteligível.

Na presente dissertação, a despeito do exposto, entendemos que ambos problemas podem vir decorrentes um do outro e, então, a possível solução deve considerar as alternativas:

- A) Eliminar o objeto;
- B) Associar um conjunto de objetos vizinhos numa única representação;
- C) Escolher uma representação simplificada que pode eliminar alguma da informação sobre o objeto, quando, em caso extremo, pode ocorrer a substituição do objeto por um símbolo;
- D) Utilizar uma representação que exceda a área efetivamente ocupada pelo objeto, o que poderá consistir numa "ampliação local", alterando a escala ou a representação do objeto específico.

A segunda alternativa é específica para tratamento de blocos, enquanto que as demais tratam de algum objeto único, porém, nada impede que o objeto a ser

tratado possa vir a ser um bloco anteriormente gerado. Em qualquer das alternativas deve-se analisar a densidade de informações, se esta é elevada ou não, e da possibilidade de já ter sido reduzida como decorrência da aplicação anterior de uma dessas alternativas.

Obviamente, o fato do tipo de informação a representar não corresponder a objetos com dimensões e localizações precisas implica maior flexibilidade na disposição dos elementos gráficos. Contudo, ainda devem ser observados problemas de densidade de informação e de dimensão mínima para a qual cada representação é inteligível.

Retornando ao trabalho de CARMO & CUNHA (1998), temos que além dos problemas de perceptibilidade da informação podemos considerar as situações em que se pretende realçar informação à custa de critérios baseados no seu conteúdo. Isto é, interessa dar mais relevância a alguma informação em relação a outra. Neste caso a representação a usar deve depender também do interesse atribuído a cada objeto, o qual poderá ser quantificado à custa de funções de grau de interesse.

As funções de grau de interesse, além de fornecerem um critério para a redução do volume de informação, podem induzir a escolha de representações diferentes consoante o grau de interesse. Assim,

"as funções de graus de interesse têm por objetivo atribuir a cada elemento de informação valor que quantifique o interesse do utilizador em visualizar esse elemento dada uma determinada tarefa (foco de interesse)."

EICK & WILLS (1993) já haviam apresentados uma aplicação simples de funções de graus de interesse, visto que, no trabalho destes, o algoritmo utilizado para a representação dos nós e ligações em uma rede de comunicação de dados baseia-se em três questões:

- A) A área de cada nó é proporcional ao número de emails enviados ou recebidos;
- B) A cor do nó representa a função do mesmo, no exemplo, azul para staff técnico, rosa para staff técnico de outros departamentos, verde para gerentes de departamento, etc.;

- C) A cor da ligação representa o fluxo de transações entre dois nós, no exemplo, existe uma escala de cores degrade do azul para o vermelho, passando por verde e amarelo, representando, nessa escala, "poucas mensagens" até "muitas mensagens".

O exemplo corrobora a generalização feita por CARMO & CUNHA (1998), onde se considera a utilização de funções de grau de interesse também no caso em que não se especifica um foco de interesse e se conhece a importância inicial em cada ponto ou, mais genericamente, uma “função de importância”. A função de grau de interesse coincidirá, neste caso, com a importância.

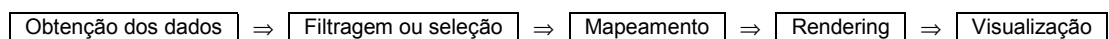


FIG. 2.5: Etapas do modelo de visualização (CARMO & CUNHA, 1998) (ALVES et al., 2000).

Um modelo de referência para visualização científica descreve, para o processo de visualização, três fases distintas entre a obtenção e tratamento de dados e a representação da imagem (FIG. 2.5) (CARMO & CUNHA, 1998) (ALVES et al., 2000): filtragem ou seleção, mapeamento e “rendering”. Ocorre, porém, que autores diferentes identificam fases intermediárias ou complementares às citadas, por exemplo, o processo de interação, complementar à exibição da imagem, que inclui o tratamento do usuário com o processo de visualização, ou a abstração, anterior ao mapeamento, que também é uma etapa de filtragem ou seleção.

A filtragem ou seleção seleciona o conjunto de dados a visualizar. Podemos entender a filtragem como um processo que não só elimina alguma informação, mas também pode alterar ou criar nova informação, por exemplo, construindo novas variáveis como função de variáveis já existentes.

CARMO & CUNHA (1998) diferenciam os processos de filtragem e de seleção, pois afirmam que podem ocorrer em diferentes fases do processo de visualização, sendo a mais importante, esta denominada de filtragem, a fase que ocorre posteriormente à obtenção, onde conhecendo-se os critérios semânticos estabelecidos pelo usuário filtra-se os dados pelo uso de funções de grau de interesse, escolhendo-se um conjunto de variáveis, restringindo-se o domínio de variação de algumas variáveis, e, como citamos, através da criação ou alteração de variáveis.

Contudo, considera-se a existência da seleção em outros pontos do processo de visualização. A seleção é um tipo de filtragem porém deve cuidar de escolher o conjunto de dados a tratar, relacionando-o com o tipo de representação que se pretender apresentar, ou seja, é um critério mais “bruto”, buscando separar um conjunto de dados, com menos significado lógico do que o processo de filtragem, que baseia-se em regras.

Na fase de mapeamento faz-se a associação entre os dados e as representações gráficas, também permitindo a ocorrência de seleção. Por exemplo, se o conjunto de variáveis a representar for muito elevado, o número de representações distintas poderá não ser suficiente para representar todas as variáveis, assim algumas variáveis terão de ser eliminadas. Trata-se de um processo de filtragem condicionado, em parte, pelas limitações do software gráfico.

Outro tipo de seleção que pode ocorrer no processo de mapeamento está relacionado com a escolha de representação de acordo com a escala de representação: se a escala for muito pequena, alguns elementos não terão uma representação inteligível e não serão representados. Esta filtragem é imposta pelas limitações da saída gráfica, em particular a resolução.

No processo de mapeamento, a associação de elementos gráficos à informação pode consistir na atribuição de um conjunto de representações e na definição de critérios de escolha da representação a usar na visualização. Os critérios estabelecidos podem ser de natureza semântica ou condicionados por características das saídas gráficas como, por exemplo, a escala em que é feita a representação.

Como vimos, no processo de *rendering* é gerada a imagem, onde também pode ocorrer a seleção, ainda segundo CARMO & CUNHA (1998), que seria uma filtragem relacionada com o controle de densidade de informação. Por exemplo, é sugerido o controle da densidade de informação através da "quantidade de tinta" utilizada no sistema representado. Isto é, a partir de certo valor para a razão entre número de *pixels* preenchidos e número total de *pixels* não são desenhados mais elementos do sistema.

Esta filtragem, pode ocorrer automaticamente por limitações da plataforma utilizada, por exemplo, pela limitação dos recursos gráficos, a saída gráfica, ou

manualmente. Isto é, quando o usuário considerar que a informação mostrada já é suficiente, ou chegou no limite de inteligibilidade, pode parar o processo de *rendering*, caso a aplicação permita. Este processo de filtragem é condicionado pelas limitações dos recursos gráficos, mas baseado em critérios de inteligibilidade. Além disso, se a informação for desenhada por ordem decrescente do seu grau de interesse, será eliminada informação menos relevante.

Com relação ao modelo descrito, CARMO & CUNHA (1998), consideram que diferentes modelos de visualização podem distinguir-se, não só por uma visualização particular para determinada estrutura de informação, como também pela existência ou não de mecanismos para filtragem de informação e de escolha da representação entre possíveis mapeamentos preestabelecidos.

O modelo proposto atua durante a fase de seleção, eliminando informação de acordo com a função de grau de interesse definida previamente, e na fase de mapeamento escolhendo a representação adequada não só em termos de escala de representação, mas também atendendo ao grau de interesse da informação.

Cabe discutir se o critério proposto para determinar a densidade da informação deveria ser ampliado, ou aplicado em conjunto, aos objetos representados e não ao sistema como um todo. Pois, um único objeto de altíssimo valor na função de grau de interesse poderia comprometer a representação do sistema, elevando à níveis de abstração pouco informativos.

Assim, se primeiro fosse feita a uma distribuição dos *pixels* disponíveis em quantidades proporcionais aos objetos a serem representados, associados a grupos de graus de interesse, talvez seria obtido uma distribuição da “quantidade de tinta” mais harmoniosa. Obviamente, que o critério relacionado ao sistema como um todo não deve ser descartado, mas, ainda atuar como um delimitador da quantidade de *pixels* preenchidos.

Sob outro aspecto, essa nova proposta pode também descaracterizar a visualização do sistema e, também comprometer sua inteligibilidade, caso provoque a representação de muitos elementos com pouquíssima quantidade de tinta.

Em HE (1999) é apresentado um visualizador para um simulador de rede de comunicação de dados, ou seja, uma implementação de uma camada (*front-end*)

que permite visualizar e entender mais facilmente os dados gerados por um simulador de rede e o comportamento da rede simulada. Para um simulador é de crucial importância o desenvolvimento de visualizadores e, este trabalho visa atender especificamente um simulador proprietário desenvolvido pela Lucent Technologies, sendo a interface visual implementada com o uso da linguagem Java.

Uma ferramenta como essa permite simular as operações e o estado de uma rede de comunicação de dados, fazendo com que a relação custo x benefício seja extremamente vantajosa no trabalho de projetistas e gerentes, pois facilita a avaliação do comportamento da rede em diversos cenários através da alteração de seus parâmetros, tais como topologia², roteamento e tráfego, sendo possível representar uma rede existente, simular diferentes tipos de falhas em nós e ligações e observar seus impactos, analisar o efeito do controle de gerenciamento da rede e otimizar seu projeto.

Segundo o autor, a principal contribuição foi o desenvolvimento de um framework genérico que provê as ferramentas necessárias para o projeto de interfaces de visualização de rede para programadores que desenvolvem sob o paradigma de orientação a objetos e ambientes web. O framework parte do princípio de abstrair os detalhes funcionais que operacionalizam as vistas de uma rede, encapsulando-os em procedimentos que, por sua vez, são agrupados em conjuntos de rotinas, denominados blocos construtores ("building blocks"). Os blocos mais importantes incluem as operações e procedimentos para manipulação e tratamento dos *dados da rede* ("network data"), *operações interativas* ("interactive operations"), *visualização das ligações* ("view linking"), *"rendering"* e *"glyph design"*³.

O bloco construtor de dados da rede permite a aquisição, o armazenamento e a comunicação dos dados da rede. Tradicionalmente, uma rede é definida como um conjunto de nós e ligações, e para tratar casos mais gerais foi adotada a representação de dados baseada em tuplas, onde uma rede é definida como um

² Por "topologia" entende-se, da geometria, a relações do posicionamento dos elementos em um espaço não quantitativo. Generalizando para teoria de grafos, a edição interativa da topologia da rede pode significar a alteração do traçado do grafo.

³ A palavra "glyph" significa uma imagem que possui um significado ou representa um recurso, ou seja, um símbolo, como, por exemplo, um ícone. No contexto, "glyph design" significa projetar,

conjunto S de n -tuplas: (a_0, a_1, \dots, a_n) , $a_i \in S$, $n \geq 0$. Cada dado de entrada (uma tupla) em uma rede é associada com atributos estáticos ou dinâmicos, como, por exemplo, o volume de tráfego corrente em um switch. Esse bloco construtor implementa algumas operações básicas (procedimentos) para manipulação de dados, incluindo criação e deleção de entradas, recuperação e modificação de informações e envio e recepção de mensagens entre blocos construtores.

O bloco construtor de operações interativas implementa três classes de operações: *seleção*, *controle de apresentação* e *operações com dados*. O propósito da seleção é identificar o conjunto de entidades apresentadas que será objeto de procedimentos futuros. Os procedimentos de controle de apresentação são utilizados na tela considerando cada elemento individualmente. Esses procedimentos são classificados como *controle visual*, que permitem a manipulação das propriedades de brilho, visibilidade e glyph, e *controle de coordenadas*, para tratamento da escala e da localização, na verdade, da movimentação do objeto na tela.

Os procedimentos de operações com dados são usados para modificar os dados subjacentes da rede, através dos quais pode-se alterar a apresentação visual da rede, incluindo a edição interativa da topologia, através de criação, remoção, cópia e colagem, assim como a modificação dos dados dos atributos. As operações com dados implementados neste bloco construtor envolvem as operações correspondentes no bloco (classe) de dados da rede.

Além disso, foi criado um novo procedimento de controle, chamado *agregado* ("aggregate"), cuja a idéia é agregar recursivamente, através de procedimento de intercalamento ("merge"), os elementos da rede, tomando por base seus tipos, e criar "elementos virtuais" agregados com os atributos da rede definidos pelo usuários.

As características do bloco construtor de visualização das ligações decorre da própria definição de HE (1999) para SVR (seção 1.2), como sendo um conjunto de vistas das diferentes perspectivas do estado de uma rede. Assim, todas as vistas são devidamente "ligadas" através de seus blocos construtores, baseado nos procedimentos interativos, tais como, seleção, foco e edição interativa.

desenhar, construir uma imagem ou, até mesmo, atribuir uma imagem, para que esta represente um objeto do sistema visualizado.

Quando um procedimento é executado em uma das vistas suas informações e seus parâmetros são, então, passados para a *máquina de visualização de ligações* que, baseado no tipo do procedimento, envia uma *mensagem de ligação* para todas as vistas correspondentes, ou seja, implementa o princípio reativo de orientação a objetos.

Em outras palavras, o princípio reativo está presente quando um método opera sobre um objeto de uma determinada classe, provocando uma alteração, e essa modificação reflete automaticamente sobre objetos pertencentes hierarquicamente à classe em questão e que estejam relacionado com a alteração. Normalmente, a alteração provocada nos objetos envolvidos se dá através de trocas de mensagens, aqui chamadas de mensagens de ligação. Uma mensagem de ligação pode acionar o mecanismo de ligações em três níveis:

- A) Nível local: Todas as operação são restritas à visualização corrente, sem propagar-se para outras vistas;
- B) Nível apresentação ("display level"): As operações são propagadas para todas as outras vistas que estão apresentando o mesmo dado;
- C) Nível de dados: Similar ao anterior, entretanto não permite a interação do usuário.

2.3 CONCLUSÕES

Visualização abrange diversas áreas, sendo algumas inicialmente não percebidas, como por exemplo, recuperação de dados, validação de sistemas, relação entre objetos, projeto de sistemas distribuídos, além de outras áreas, que podem ser consideradas clássicas, citadas ao longo do texto. Destaca-se, também, de forma histórica e tradicional, a grande utilização como ferramenta de apoio ao aprendizado e no ensino à distância, esta último, com maior ênfase nos últimos anos.

As publicações e as implementações disponíveis relacionadas neste capítulo e que tratam de SVR, apresentam ferramentas e abordagens específicas para

determinados problemas, focalizando questões e assuntos determinados. Ou seja, nenhum dos trabalhos, aos quais tivemos acesso, propõem uma solução generalizada para visualização de redes.

Especificamente para visualização de grafos e algoritmos, existem vários trabalhos e ferramentas para “desenho” e edição de grafos, que normalmente manipulam poucas informações sobre as entidades que os compõem, e de animação, que normalmente ilustram o funcionamento de algoritmo e são utilizados em processos de ensino e aprendizagem.

Sob o aspecto de composição do modelo de redes, apesar da definição formal de redes, $N = (G, I)$, incluir apenas informações sobre pontos e ligações, foi percebida a necessidade de tratar informações globais sobre a rede, a partir da utilização do modelo de rede para representar um sistema, processo ou serviço, ou seja, uma rede de serviço. Assim, os elementos que compõem uma rede de serviço devem deixar de ser somente elementos representados por vértices e arestas, e incluir também um macroelemento: a própria rede ou grafo.

Além disso, a existência desse novo elemento também se justifica a partir da determinação do procedimento de “tratamento de bloco”. Um bloco é a especificação de uma parte da rede ou grafo, podendo representar um subgrafo ou uma subrede, ou seja, qualquer procedimento que venha a tratar um bloco, na verdade, pode vir a tratar o elemento rede/grafos como um todo.

Decorre daí o problema geral de representação de redes, em termos de visualização: quais, e como, os objetos podem, ou devem, ser agrupados para facilitar ou permitir a apresentação, principalmente quando existe um grande número de pontos ou ligações. Assim, para tratar tal problema, os procedimentos de controle de apresentação trazem alguns métodos padrões de redução, incluindo transformação, merge, “olho de peixe” e filtros interativos, por exemplo, subrede e foco.

O grande volume de dados também pode ser tratado de forma não interativa, aliás, como é mais comum, quando existe a filtragem no processo de representação da imagem. De acordo com o número de citações, o procedimento mais utilizado e de maior efetividade é a filtragem, que ocorre fundamentalmente com base em critérios semânticos.

Outro processo de filtragem, conhecido como seleção, não se prende a critérios semânticos, podendo ocorrer, por exemplo, por limitações de recursos físicos, tais como, as limitações das saídas gráficas.

Percebe-se um senso comum entre autores quando, de forma subliminar, admitem que a existência de grandes volumes de dados no estudo de redes possa ser uma situação comum e normal. Sob esse aspecto, justifica-se, primeiramente, a importância do papel das informações de uma rede, como objeto de estudo e de análise da mesma e, além disso, se ressalta a grande massa de dados e informações que se pode obter, seja de forma analítica, ou seja, pontualmente, ou estatística, com informações agrupadas ou já interpretadas.

Com o foco voltado para a representação de redes de serviços, quando a visualização assume seu papel de ferramenta facilitadora do entendimento do sistema representado, se vê que os autores buscam transpor obstáculos que impeçam uma rápida e precisa leitura através de filtros e seleções dos dados, cores, tamanhos, convenções e símbolos na representação de objetos, e a disposição do desenho sobre um contexto visual de conhecimento prévio para o usuário. Contudo, estranhamente, não procuram criar tal facilidade disponibilizando desenhos e traçados para novas disposições visuais da localização dos pontos da rede, que poderiam, por exemplo, significar uma nova disposição geográfica, localização física ou, até mesmo, uma alteração da topologia da rede, se for considerada apenas as relações de posicionamento.

Analisando em conjunto os trabalhos apresentados, se constata que trabalhos na área de visualização dividem-se em dois grupos: estudar, apresentar ou simular o comportamento dos dados ou o comportamento dos procedimentos. Os trabalhos relacionados com o comportamento dos procedimentos são mais voltados para a computação básica e foram elaborados no início dos estudos da visualização. Em contrapartida, os trabalhos relacionados com o comportamento dos dados, são voltados para computação aplicada e, em geral, são mais recentes.

Em termos gerais, como benefício em todas as áreas nas quais se vê o uso da visualização, mostra-se sua utilização como parte das ferramentas voltadas ao entendimento e leitura do que se quer representar. A necessidade de visualização

é premente, e um diferencial, em qualquer intenção de se buscar meios mais fáceis de compreensão.

3 O KINEGRAPH

3.1 APRESENTAÇÃO

O software KineGraph é um trabalho existente na literatura nacional e, devido à sua importância no escopo desta dissertação, conforme exposto na seção 1.4, merece um capítulo a parte.

O KineGraph, atualmente na versão 3.0, é um ambiente para a implementação de algoritmos em grafos, sem, contudo, disponibilizar um traçador (visualizador) de grafos, visto não ser este seu objetivo. Segundo MARKENZON & VERNET (1998), os autores desta ferramenta, grafos são estruturas cuja a representação pictorial é inerente e, então, intuitivamente visualiza-se grafos a partir da construção de imagens mentais, todavia, durante a implementação de um algoritmo, necessita-se da descrição, do armazenamento e da manipulação dos dados relacionados, de forma concreta e formalmente estabelecida.

Assim, segundo os autores, o KineGraph é um conjunto de ferramentas, programas e tipos abstratos de dados, disponíveis como bibliotecas de rotinas, que permite o projeto, a implementação e o acompanhamento (“debugging”) de algoritmos em grafos de forma simples e integrada, objetivando resolver de forma unificada as seguintes questões:

- A) A descrição em alto nível de declarações específicas para grafos quando são traduzidas para declarações de linguagens de programação provocam, com certa frequência, uma grande distância entre a essência da idéia no algoritmo com o código gerado no programa.
- B) Partindo da geração de dados de entrada, seja manual ou automática, até a interpretação final dos mesmos, exige-se conversões pouco espontâneas. Por exemplo, a representação gráfica de um grafo é completamente diferente das representações computacionais usuais, tais como, matrizes, listas ou conjuntos.

- C) A tradicional abordagem para o acompanhamento da execução de programas, através da verificação de valores de variáveis, pode comprometer o entendimento da execução do programa como um todo.

Assim, o KineGraph quer prover um conjunto de ferramentas para a integração de todos os passos na solução de um problema em grafos (MARKENZON & VERNET, 1998).

A principal característica do KineGraph são suas bibliotecas de rotinas que implementam os procedimentos que possibilitam ao usuário manipular a estrutura de dados de grafos, vértices e arestas. A descrição a seguir, retirada do seu manual (MARKENZON & VERNET, 1997), apresenta com clareza a estrutura e característica do ambiente, bem como se suas bibliotecas:

“(O KineGraph) é composto por três módulos, sendo duas bibliotecas de rotinas e um programa:

- *A biblioteca de rotinas básicas, ou GrBasic - um conjunto de operações básicas sobre a estrutura grafo, para implementação de algoritmos.*
- *A biblioteca de animação, ou GrAnima - um conjunto de operações para o acompanhamento visual da execução de programas.*
- *O construtor de grafos, ou simplesmente Construtor - um programa que possibilita a edição gráfica de grafos.*

A principal característica do sistema é possibilitar o intercâmbio dos tratamentos gráfico e algorítmico dos grafos envolvidos. Assim, um grafo editado via Construtor serve como entrada para um programa qualquer, desenvolvido pelo usuário. Com as rotinas da GrBasic, este programa pode explorar as informações lógicas, os atributos do grafo e as informações gráficas associadas aos vértices e arestas no momento da edição.

A execução de um programa pode modificar características gráficas dos grafos que estiverem sendo tratados, alterando, por exemplo, cores e formas de vértices e arestas. Com

chamadas às rotinas da GrAnima, estes grafos podem ser sucessivamente exibidos a cada alteração, o que permite acompanhar visualmente o desenrolar do processo.

O ambiente KineGraph armazena um grafo em um arquivo com extensão “.gra”. Não são impostas restrições quanto ao tamanho ou número de grafos que um programa pode tratar simultaneamente. Limitações podem surgir, entretanto, decorrentes da quantidade de memória do equipamento e de peculiaridades de alocação, inerentes ao DOS e ao Turbo Pascal.”

3.2 COMENTÁRIOS E AVALIAÇÃO

Considerando a disponibilidade atual de recursos e plataformas computacionais e ainda os avanços da engenharia de software no que tange construção de interfaces, o KineGraph não deixa nada a desejar e sob a ótica do funcionamento geral e recursos apresentados, disponibilizando para o usuário os recursos necessários para a manipulação da estrutura representada, no caso, um grafo.

Conforme observado ao longo do texto, ferramentas de visualização são dependentes do ambiente, dos recursos computacionais e dos recursos disponibilizados pelas linguagens de programação, e invariavelmente um dos pontos críticos em SV é a interface disponibilizada aos usuários. Assim, por ser a visualização uma área dependente da tecnologia disponível, seus produtos tendem à obsolescência em curtos espaços de tempo.

O KineGraph foi desenvolvido criteriosamente para criar uma interface que atenda às necessidades dos usuários, a despeito dos fracos recursos disponíveis na plataforma de desenvolvimento utilizada, sem recursos gráficos que abstraíam e facilitem o tratamento de dispositivos de entrada e saída ou os eventos que traduzam as solicitações dos usuários.

No desenvolvimento do ambiente em si foi implementado uma vasta quantidade de recursos e procedimentos necessários para criar a interface de trabalho para o usuário, visto que a plataforma hospedeira não os oferecia. Por exemplo, a plataforma não oferecia uma interface gráfica nativa e esta foi desenvolvida dentro do KineGraph.

Desta forma, comparado a projetos atuais, observa-se no KineGraph que a gama de procedimento e recursos disponibilizados são oferecidos em quantidade e função adequadas, mas que foram, por necessidade, construídos sob complexos conceitos que extraem o máximo dos recursos do sistema de computação hospedeiro.

Os próprios autores, obviamente, sabem das restrições que o ambiente impôs, que em alguns casos limita a abrangência do KineGraph e observam a necessidade de migração das plataformas de desenvolvimento e execução quando, em MARKENZON & VERNET (1998), sugerem uma nova versão do KineGraph, utilizando a linguagem C, sobre o ambiente operacional Unix com interface gráfica X Windows. Enquanto que a versão ora tratada, KineGraph 3.0, foi desenvolvida na plataforma Turbo Pascal 7.0, sobre DOS.

Sob o ponto de vista de conceitos de linguagens e técnicas de programação utilizados na implementação propriamente dita, observa-se que o tipo abstrato de dados implementado para a manipulação dos elementos de um grafo, foi feito de forma conceitual, sem utilizar as construções de abstração da linguagem fonte. Fato que, de forma alguma, depõe contra o software. Na verdade, sob o ponto de vista de aprendizado torna-se um interessante estudo de caso.

Além disso, ao considerarmos uma linguagem de programação como sendo a notação formal para descrever a execução de algoritmos em computador, sendo composta de, basicamente, sintaxe e semântica (SILVA & ASSIS, 1988), a implementação da GrBasic, apresenta-se com um forte propósito de ser um protótipo de uma linguagem de programação específica para manipulação de grafos, apesar de não ter sua especificação formal, na versão ora apresentada.

É interessante observar que para definir o escopo do KineGraph, os autores usam firmemente os fundamentos básicos (seção 1.1) de visualização, em "... intuitivamente visualizam-se grafos a partir da construção de imagens mentais...", e de animação de algoritmos, afirmando a distância entre descrições de alto nível,

que seria o modelo do problema na linguagem do usuário, e as declarações de uma implementação computacional do problema modelado.

4 O KITNET

4.1 APRESENTAÇÃO E OBJETIVOS

Dentro da proposta de trabalho encontra-se o desenvolvimento de um protótipo de ferramenta de visualização de redes de serviço, doravante denominado KitNet, com a intenção de validar o levantamento e o estudo realizado, bem como as conclusões obtidas até o momento.

O desenvolvimento do KitNet não se baseou no software KineGraph, já apresentado, mas, com autorização dos autores, foi selecionada e tomada como ponto de partida uma das bibliotecas de rotinas, a GrBasic, visto que o KineGraph possui bibliotecas criteriosas e bem desenvolvidas com algoritmos para a manipulação de estruturas de dados e para a representação e apresentação de grafos.

O KitNet é um protótipo de SV, ou seja, uma ferramenta que implementa as características desejáveis ou necessárias de um SV, obtidas a partir da análise feita no segundo capítulo, quando foram identificado os conjuntos fundamentais (ver anexo “B”) de procedimentos, como os instrumentos para manipulação dos objetos representados, e o de elementos que compõem a rede de serviço, que são os objetos representados. Contudo, como protótipo, não virá a cobri-los em sua totalidade.

O protótipo apresenta-se abrangente sob o ponto de vista de tipos de redes que manipula, pois trata as mesmas como grafos, face as estruturas de armazenamento e representação que utiliza, estando em consonância com o conceitos de redes de serviço, apresentado na seção 1.2. Como consequência, esta abordagem disponibiliza todos os recursos do protótipo indiferentemente da rede representada, não particularizando o tratamento para nenhum tipo de sistema.

A proposta que embasa a construção do KitNet busca o desenvolvimento de uma ferramenta que atenda quesitos mínimos de um SV, que permita ao usuário

interagir com o sistema representado, pretendendo tornar-se um recurso facilitador da “leitura” da rede, como consequência, do entendimento do sistema.

Assim, como objetivo específico, o KitNet vai, em primeira instância, atuar sobre o traçado, o desenho e a apresentação de uma rede, isto é, será uma ferramenta que permitirá a representação e a manipulação visual de um sistema.

Em um segundo momento, é sabido que o objetivo almejado somente será satisfeito se o protótipo oferecer um conjunto de recursos que esteja alinhado com as tecnologias e recursos computacionais atuais e um conjunto de procedimentos que satisfaça necessidades inerentes à visualização, como, por exemplo, recursos que permitam criar, editar, alterar, armazenar e ler um determinado sistema ou rede. Entretanto, a proposta de funcionamento do KitNet é ser o mais simples e direto o possível, tendo como meta que seu modo de operação seja, para o usuário, tão próximo da intuição quanto possível.

O KitNet não pretende, de forma alguma, gerar uma nova versão do KineGraph, mas, em parte, quer cobrir uma das lacunas deste software, no que tange a defasagem tecnológica com relação a interface e ao ambiente computacional hospedeiro do citado software. Conforme discutido em 3.2, tal defasagem é extremamente comum na área, assim, a definição correta da plataforma para o desenvolvimento do KitNet e da plataforma hospedeira do código executável contribui de maneira significativa para a expectativa de vida útil do software e para seu uso efetivo.

4.2 DESCRIÇÃO DAS PLATAFORMAS E DO PROTÓTIPO

A plataforma de desenvolvimento irá influenciar na metodologia, técnica e estilo de desenvolvimento, nos possíveis recursos a serem oferecidos pelo software e na correta implementação das soluções aos problemas levantados e tratados. Por sua vez, o ambiente de execução influencia diretamente no grau de dificuldade para a implementação das soluções, nas restrições impostas pelo conjunto software e hardware quanto aos recursos oferecidos, e na proposta geral e escopo do software.

Em adição, as plataformas de desenvolvimento e de execução não devem simplesmente contar com os recursos tecnológicos mais recentes, mas ainda oferecer resposta aos itens que influenciam cenários futuros, e, de forma geral, serem estáveis e robustas.

Apesar de o trabalho ter sido iniciado com base na plataforma “Intel - MS Windows - Delphi”, respectivamente, hardware, sistema operacional e linguagem, logo no princípio da implementação optou-se por trocar o ambiente operacional de Microsoft Windows 98 para GNU/Linux e a linguagem de Delphi para Kylix, mantendo a plataforma de hardware.

Com mais detalhes, a plataforma computacional utilizada foi um microcomputador dotado de processador Intel Pentium II, frequência de 400 MHz, com 128 MB de memória, tendo como sistema operacional o GNU/Linux, com kernel versão 2.2.16-22, distribuição Red Hat versão 7.0. Como ambiente de desenvolvimento foi utilizada o Kylix 1.0, versão Open Edition, disponibilizada gratuitamente através o site do fabricante⁴. Já no fim do desenvolvimento do protótipo foi migrado para o Kylix 2.0, também na versão Open Edition.

O desenvolvimento do KitNet iniciou-se com o porte da biblioteca de rotinas básicas, a GrBasic (seção 3.1), para o ambiente adotado. O porte do código fonte de uma linguagem procedimental em blocos, de paradigma estruturado, como o Pascal, que apresenta como propriedades, por exemplo, “tipagem” forte e estática e o “binding” podendo ser feito em tempo de compilação ou execução, dependendo da declaração das rotinas e de seus argumentos, propicia dificuldades inicialmente não percebidas, e quase imprevisíveis.

Em resumo, a nova linguagem a ser utilizada, Kylix, foi projetada sob outro paradigma, no caso, orientação a objeto, que apresenta propriedades diferentes ou até inversas, por exemplo, “tipagem” fraca e dinâmica e o “binding” em tempo de execução, necessárias para atender às características de sobrecarga de operadores e outras formas de poliformismo.

Estabelecidos os conceitos de funcionamento, a partir dos objetivos citados, e as plataformas, de onde obtêm-se os recursos para implementação dos conceitos, é definido o escopo do protótipo, ou seja, onde o protótipo deve atuar, ou ainda, o que deve ser disponibilizado para o usuário para satisfazer suas

⁴ <http://www.borland.com/>

necessidades. O que, como efeito colateral, gera ou expõe novas restrições, além das restrições inerentes à escolha das plataformas.

Basicamente, existem dois tipos de objetos que são manipulados pelo software. São eles: o arquivo de armazenamento da estrutura de dados da rede em meio não volátil, e a estrutura de dados da rede para manipulação interativa, ou seja, a estrutura manipulada durante a interação do usuário com o protótipo.

Nesse ponto o KitNet utiliza a biblioteca GrBasic como uma camada de nível inferior de implementação, para a manipulação da rede como um grafo. Ou seja, o KitNet cria uma estrutura para armazenar informações sobre o grafo, ou informações da rede, e associá-las ao grafo armazenado a partir da estrutura definida na GrBasic, tomando a definição de rede de serviço em 1.2, $N = (G, I)$.

Partindo dos dois objetos citados, serão apresentadas resumidamente a seguir as formas de manipulação oferecidas aos mesmos, tendo em vista que em muitos casos podem confundir-se com funções disponibilizadas aos usuários. Assim, para tratar o objeto em arquivo, o software disponibilizará procedimentos de leitura e gravação do arquivo no formato KitNet, importação e a conversão de arquivos em formato texto, em um padrão definido, e de arquivos no formato KineGraph.

Para manipulação do objeto em memória, o KitNet deve disponibilizar ao usuário procedimentos de criação de uma rede com geração aleatória e, partindo de uma estrutura vazia, inclusão, remoção e movimentação de pontos e ligações da rede, localização de um ponto, seleção de um ponto, navegação entre instâncias de uma rede, que será explicado posteriormente, e alguns traçados para apresentação da rede com diferentes “lay-outs”.

Sob a ótica de funcionamento, com a qual o KitNet será descrito nas próximas seções, para manipulação dos objetos anteriormente explicitados, o protótipo habilita ao usuário fazer:

- A) A entrada de dados da rede por geração aleatória automática ou interativa ou, ainda por importação de dados;
- B) A edição da rede;
- C) A visualização da rede, através de traçados automáticos.

Ainda sob este prisma, e buscando simplificar a interação entre o usuário e protótipo, as funções estarão disponíveis na interface gráfica através de menus reduzidos e concisos, pois o protótipo priva pela simplicidade em sua operação, buscando a funcionalidade em prol da estética.

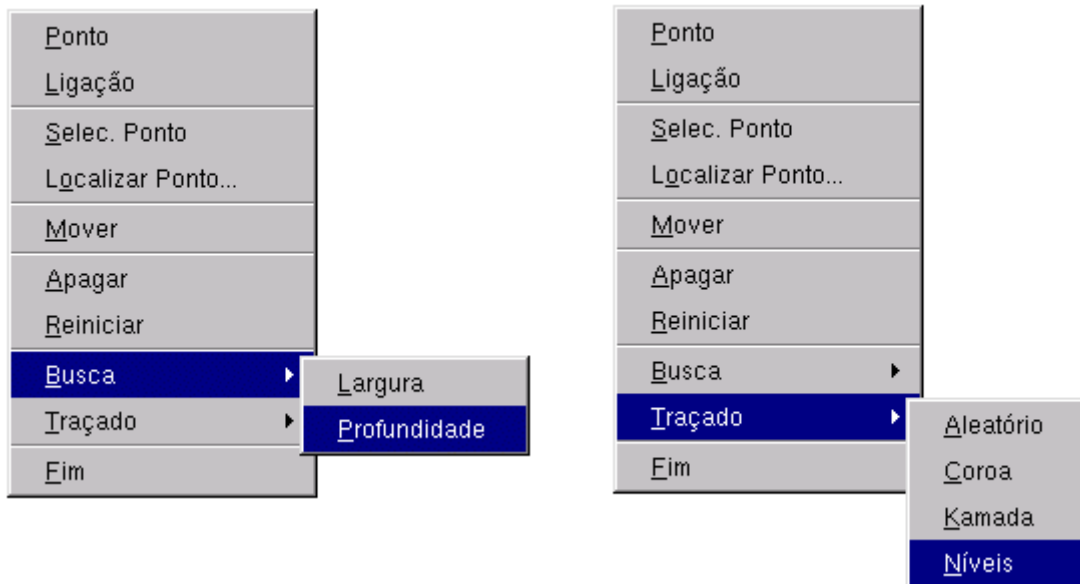


FIG. 4.1: Menu de procedimentos.

O projeto e a implementação do protótipo seguiu o princípio de simplificar ao máximo os elementos da interface e, então foram criados apenas três menus distintos que agrupam os recursos e procedimentos de acordo com a funcionalidade dos mesmos.



FIG. 4.2: Menu "Rede".

O menu de procedimentos (FIG. 4.1) constitui-se de um conjunto de opções que permite ao usuário atuar sobre a rede, para fazer a criação interativa, a edição, o desenho, o traçado, etc.. Especificamente, neste menu são disponibilizados os procedimentos de inclusão de pontos (nós), de ligação

(arestas), de remoção de pontos ou ligações, de movimentação de pontos e suas ligações incidentes, e de seleção de pontos. Ainda permite que se reinicie todo o trabalho de criação ou edição, bem como que se finalize o uso do software. Além disso disponibiliza os traçados e buscas, que serão descritos posteriormente.

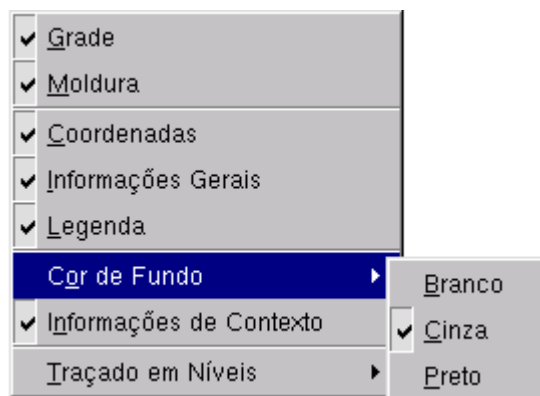


FIG. 4.3: Menu "Ver".

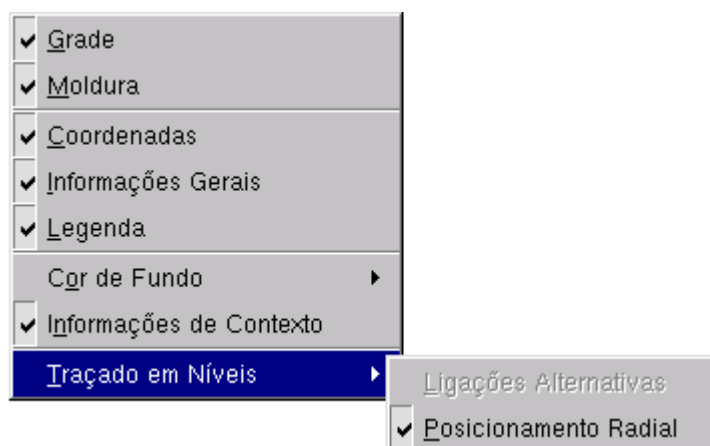


FIG. 4.4: Menu "Ver".

Os dois outros menus, disponíveis a partir da barra de menus, que está localizada na barra superior da janela, são o menu "Rede" (FIG. 4.2), que será explicado mais adiante, e o menu "Ver" (FIG. 4.3 e 4.4), que agrega as "Informações de Contexto" da interface de visualização, ou seja, agrega alguns elementos que tornam a interface mais conveniente ao uso.

Através deste é possível mostrar ou esconder a legenda, a moldura, que delimita a área útil da tela do KitNet, a grade, que auxilia no posicionamento e localização dos elementos da rede, as coordenadas do posicionamento do cursor, as informações básicas sobre a rede representada e ainda alterar a cor de fundo

da tela, adequando-a a situações de menor ou maior intensidade de luz⁵. Neste menu, como pode ser visto na FIG. 4.4, também encontram-se duas informações de contexto mais específicas, associadas às buscas ou a exibição da rede quando desenhada em níveis, são elas: “Ligações Alternativas” e “Posicionamento Radial”. Essas opções serão abordadas posteriormente, porém a primeira opção permite esconder ou visualizar as arestas de retorno obtidas em uma busca ou traçado em níveis, e a segunda opção, se habilitada, muda o desenho do traçado em níveis, distribuindo os pontos radialmente em relação à origem (raiz).

O menu “Rede” (FIG. 4.2) agrega as opções que manipulam os arquivos de armazenamento dos dados da rede. Disponibiliza a criação da rede por geração aleatória e automática, e a obtenção da rede, através da leitura de arquivos no formato KitNet ou importação de arquivos texto, com formato específico, ou KineGraph. Neste menu também encontram-se as opções para reiniciar a sessão de uso do software, retornando ao estado inicial de execução, e para a finalização da sessão de uso.

Além disso, na opção “Gravar...” podem-se arquivar os dados de uma rede criada, editada ou visualizada, e, em “Exportar...”, converter uma rede visualizada para arquivos de dados em formato texto, seguindo o padrão estabelecido, ou no formato KineGraph.

4.3 ENTRADA DE DADOS DA REDE

O KitNet permite a entrada de dados da rede de duas formas: geração aleatória ou obtenção, isto é, importação de dados. A geração aleatória da rede pode ser feita de forma interativa ou automática e, na opção de obtenção, permite-se a importação de dados oriundos de um formato padrão, em arquivo texto, ou arquivos oriundos do KineGraph.

⁵ Como ilustração dos recursos do protótipo, na composição deste texto foram utilizadas algumas figuras exibindo o fundo em preto e outras em cinza, contudo na sua maioria os exemplos e ilustrações são apresentados em fundo branco.

A criação de uma rede por geração aleatória e interativa permite que o usuário venha a desenhar sua rede, dispondo os pontos e ligações à sua conveniência, utilizando-se de recursos totalmente visuais, visto que esse procedimento utiliza a tela inicial (FIG. 4.5), que também é utilizada em todos as funções e procedimentos para manipulação das redes, inclusive os de visualização.

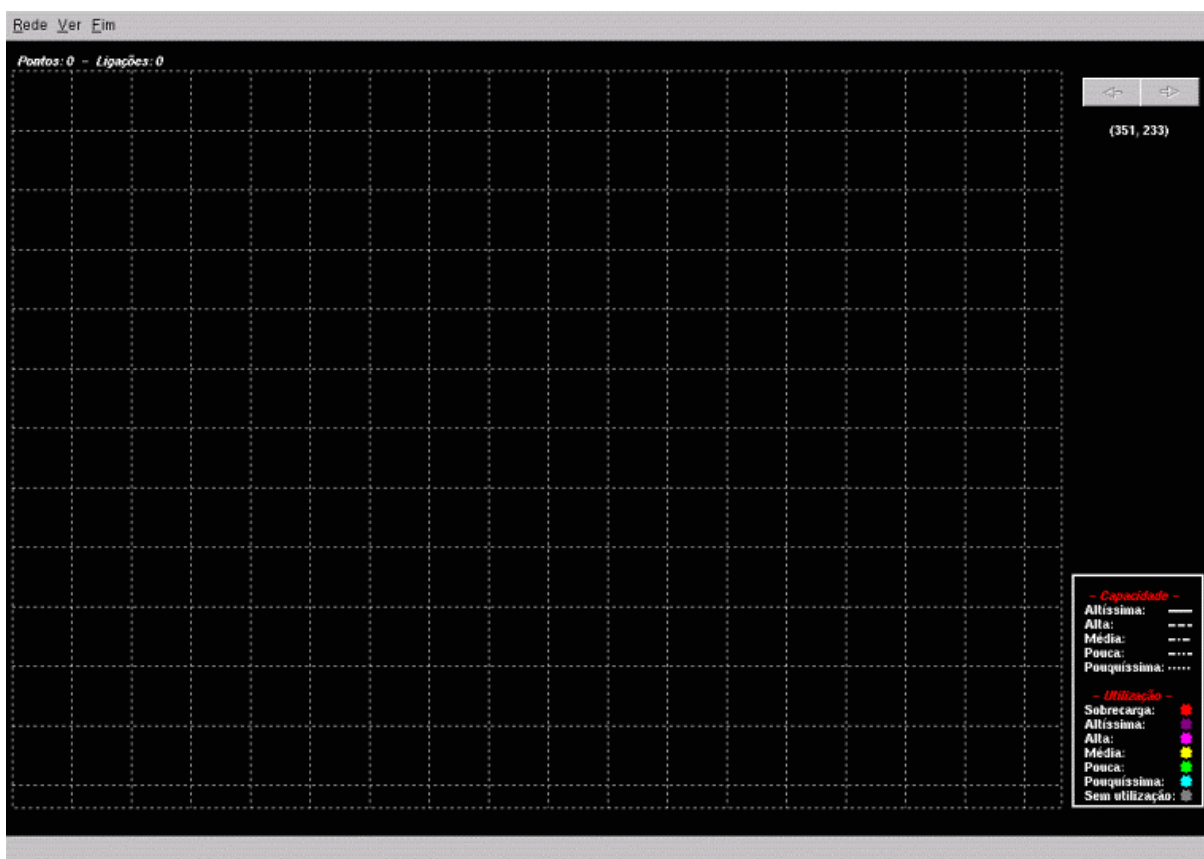


FIG. 4.5: Tela inicial do KitNet.

Pode-se entender que a criação interativa de uma rede é a edição de uma rede inicialmente vazia, ou seja, sem nenhum ponto ou ligação e, assim, o usuário seleciona a função desejada no menu de procedimentos (FIG. 4.1), disponibilizado através do botão direito do mouse. Por exemplo, para inserir os pontos da rede, basta selecionar a opção “Ponto”, disponível no menu de procedimentos, posicionar o cursor no local desejado, na área delimitada pela moldura da tela inicial, e clicar com o botão esquerdo do mouse.

Caso seja opção do usuário criar uma rede automaticamente, então deve ser acessada a opção “Criar...” no menu “Rede” (FIG. 4.2). Este procedimento

permite ao usuário personalizar a rede a ser criada com opções bastantes flexíveis, e o KitNet se encarregará de inferir, ou simplesmente “sortear”, os demais dados necessários à criação da rede.

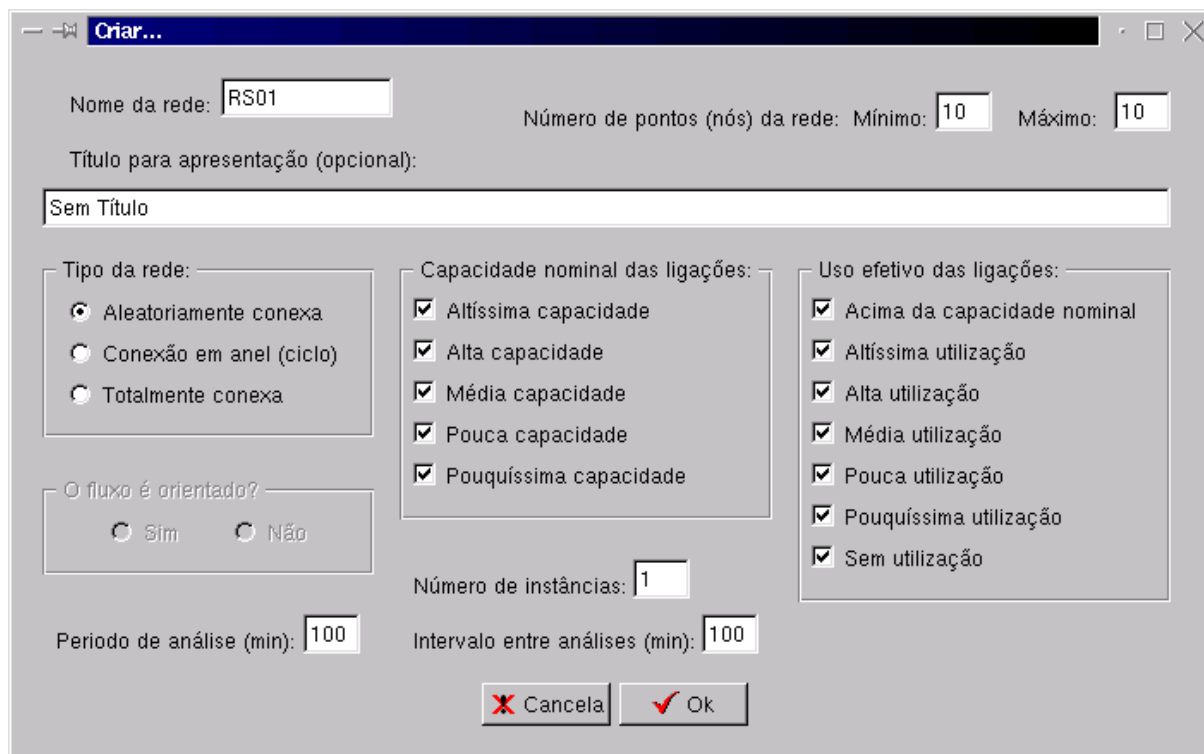


FIG. 4.6: Menu “Rede” → “Criar...” - Tela para criação automática da rede.

Como exemplo de criação de uma rede de forma automática (FIG. 4.6), inicialmente deve-se escolher um nome para a rede a ser criada, que servirá de nome base para os arquivos dos dados. O usuário também poderá inserir um título, comentário ou observação sobre a rede em “Título para apresentação”, que será usado nas “Informações gerais” quando a rede for visualizada. Em seguida, pode-se optar por um número máximo e mínimo de pontos, quando então o protótipo decidirá aleatoriamente a quantidade exata entre os valores estipulados. Caso o usuário deseje um valor exato, basta igualar os limites máximo e mínimo. Além disso, o usuário também pode decidir como será a ligação entre os pontos, optando entre aleatória, totalmente conexa ou um ciclo simples entre os pontos.

Com relação às ligações pode-se escolher a capacidade nominal das mesmas, entre altíssima, alta, média, pouca e pouquíssima, e também pode-se escolher a utilização efetiva das mesmas, entre acima da capacidade, altíssima, alta, média, pouca, pouquíssima e sem utilização.

Permite ainda que se estabeleça o número de instâncias da rede que se deseja apresentar, ou melhor, quantos conjuntos de dados da rede que serão criados e que poderão ser visualizados. O número de instâncias pode ser escolhido alterando-se os valores padronizados de número de instâncias, período de análise ou intervalo entre análises.

No escopo desta dissertação, chamam-se instâncias, ou grupos de instâncias, distintos conjuntos de dados de uma mesma rede que juntos produzem a idéia de “visões” do funcionamento da rede, que se alteram ao longo de um eixo determinado, sem, no entanto, que seja visto como um trabalho de animação ou mesmo uma forma de animação. Além disso, deve-se assumir, por mera convenção, que o eixo de deslocamento dessas vistas, ou melhor, das instâncias, representa o tempo.

Nome da rede: REDE1 Número de pontos (nós) da rede: Mínimo: 5 Máximo: 10

Título para apresentação (opcional):
Rede local de teste

Tipo da rede:
 Aleatoriamente conexa
 Conexão em anel (ciclo)
 Totalmente conexa

O fluxo é orientado?
 Sim Não

Capacidade nominal das ligações:
 Altíssima capacidade
 Alta capacidade
 Média capacidade
 Pouca capacidade
 Pouquíssima capacidade

Uso efetivo das ligações:
 Acima da capacidade nominal
 Altíssima utilização
 Alta utilização
 Média utilização
 Pouca utilização
 Pouquíssima utilização
 Sem utilização

Número de instâncias: 10

Período de análise (min): 100 Intervalo entre análises (min): 10

FIG. 4.7: Exemplo da tela para criação de uma rede de forma automática.

O uso deste recurso possibilita a obtenção de vistas de uma rede, gerando informações para testar um sistema que se quer representar e analisar como, por exemplo, gerar um conjunto de dados para um rede chamada “REDE1”, com número de pontos entre 5 e 10, aleatoriamente conexa, com ligações de capacidade nominal alta, média e pouca e com uso efetivo acima da capacidade, alta, média, pouca e sem utilização, e 10 instâncias. Neste caso, a tela de criação

apresenta-se como na FIG. 4.7 e a visualização de três de suas instâncias geradas podem ser vistas nas FIG. 4.9, 4.10 e 4.11.

Após a execução do procedimento, o KitNet gerou uma rede com 7 pontos, sorteado entre os limites máximo e mínimo, que foram dispostos de forma aleatória, e também de forma aleatória foram geradas as ligações entre os pontos, visto a opção selecionada em “Tipo de rede”.

Um grupo de instâncias caracteriza-se por manter fixo o conjunto de pontos da rede, em quantidade e localização, podendo variar os valores associado às ligações, assim como as próprias. Sua visualização é feita a partir da tela inicial, que, como observado anteriormente é utilizada para qualquer manipulação da rede, ou seja, a tela inicial muda de função à medida que for necessário, podendo ser a tela de criação interativa, a tela de edição ou a tela de visualização.



FIG. 4.8: Legenda da tela de visualização.

Na tela de visualização é apresentado o conjunto das “Informações de Contexto”, que é composto das “Informações Gerais” que agrega o “Título para apresentação”, se existir, com o número de pontos e de ligações da rede, e ainda a legenda, as setas que permitem avançar ou retroceder entre as instâncias, as coordenadas de localização do cursor, a grade e a moldura da área de visualização. Como visto, excetuado-se as setas de movimentação entre instâncias, qualquer um dos objetos que compõem a tela de visualização podem ser escondidos, conforme necessidade ou desejo do usuário, podendo desabilitá-los a partir do menu “Ver”. As setas de movimentação entre as instâncias possuem habilitação automática, ou seja, se a rede visualizada possui apenas um conjunto de dados, uma instância, então as setas são automaticamente inibidas, caso contrário, são disponibilizadas.

Para melhor entendimento do exemplo REDE1, será feita um acompanhamento das instâncias apresentadas a seguir. Na visualização da instância 6 (FIG. 4.9) vê-se como “Informações gerais” o título, o número de pontos e de ligação da instância em questão, ou seja, “Rede local de teste – Pontos: 7 – Ligações: 12”. Tomando como exemplo os pontos (nós) 1 e 2, temos, nesta instância, uma ligação representada por uma linha tracejada e de cor vermelha, isto é, significando, de acordo com a legenda (FIG. 4.8), que neste instante representa uma ligação de alta capacidade e que está sobrecarregada.

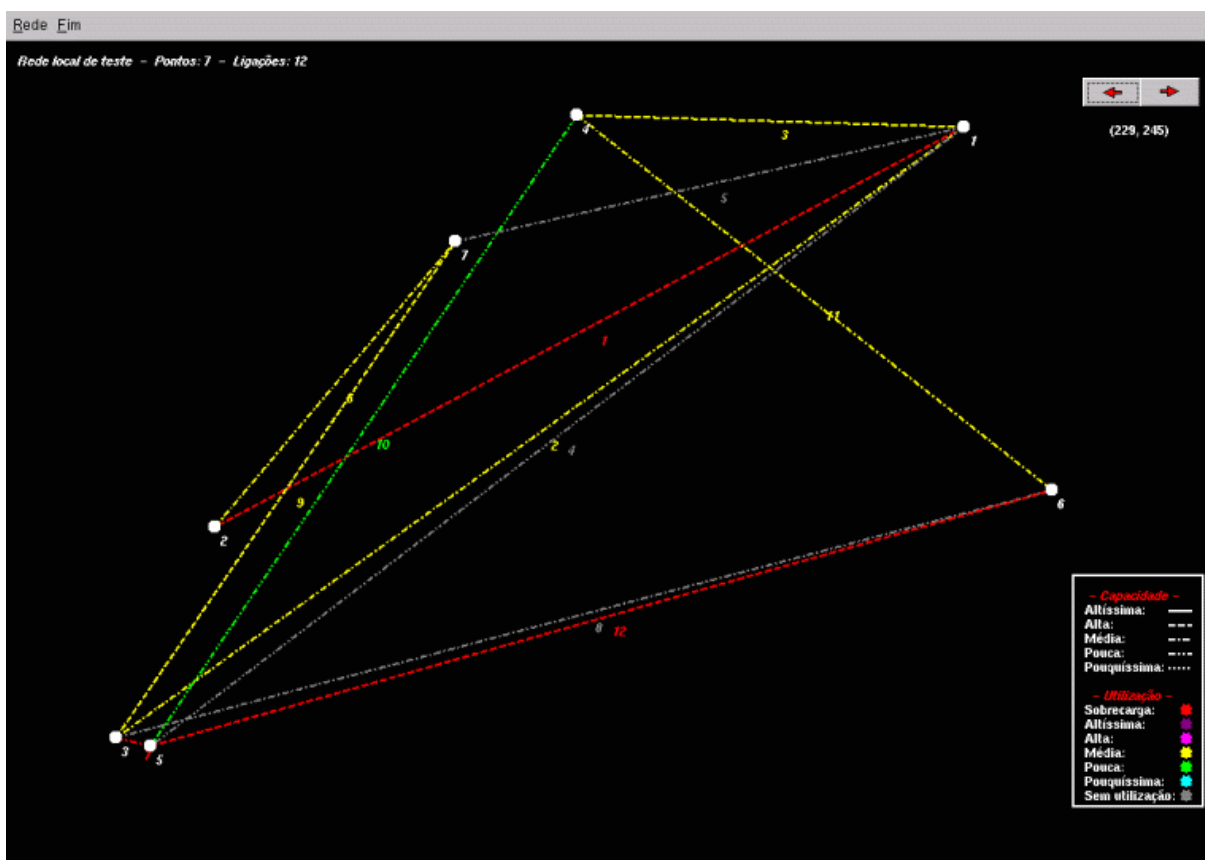


FIG. 4.9: Visualização da instância 6 de REDE1.

Verificando os mesmos pontos na FIG. 4.10, a instância 7, vê-se que a ligação deixou de existir. Como no momento anterior a ligação apresentava-se sobrecarregada, o sistema representado pode ter optado por retirar a ligação redistribuir o fluxo entre as ligações para posteriormente retornar a ligação (1, 2) redimensionada.

A ausência da ligação (1, 2) significa que a mesma não existe no momento, diferentemente da ligação (1, 3), representada nesta mesma instância, onde a

ligação existe, porém a convenção adotada nos mostra que não está sendo utilizada.

Já na instância 8, FIG. 4.11, ambas as ligações existem e, por coincidência, ambas se encontram sobrecarregadas, sendo que, no momento, a ligação (1, 2) se apresenta com média capacidade e a ligação (1, 3) manteve-se com alta capacidade. Também deve-se notar, como especificado, que por se tratar de instâncias de uma mesma rede, os pontos são os mesmos e, portanto sem alteração da localização de cada um.

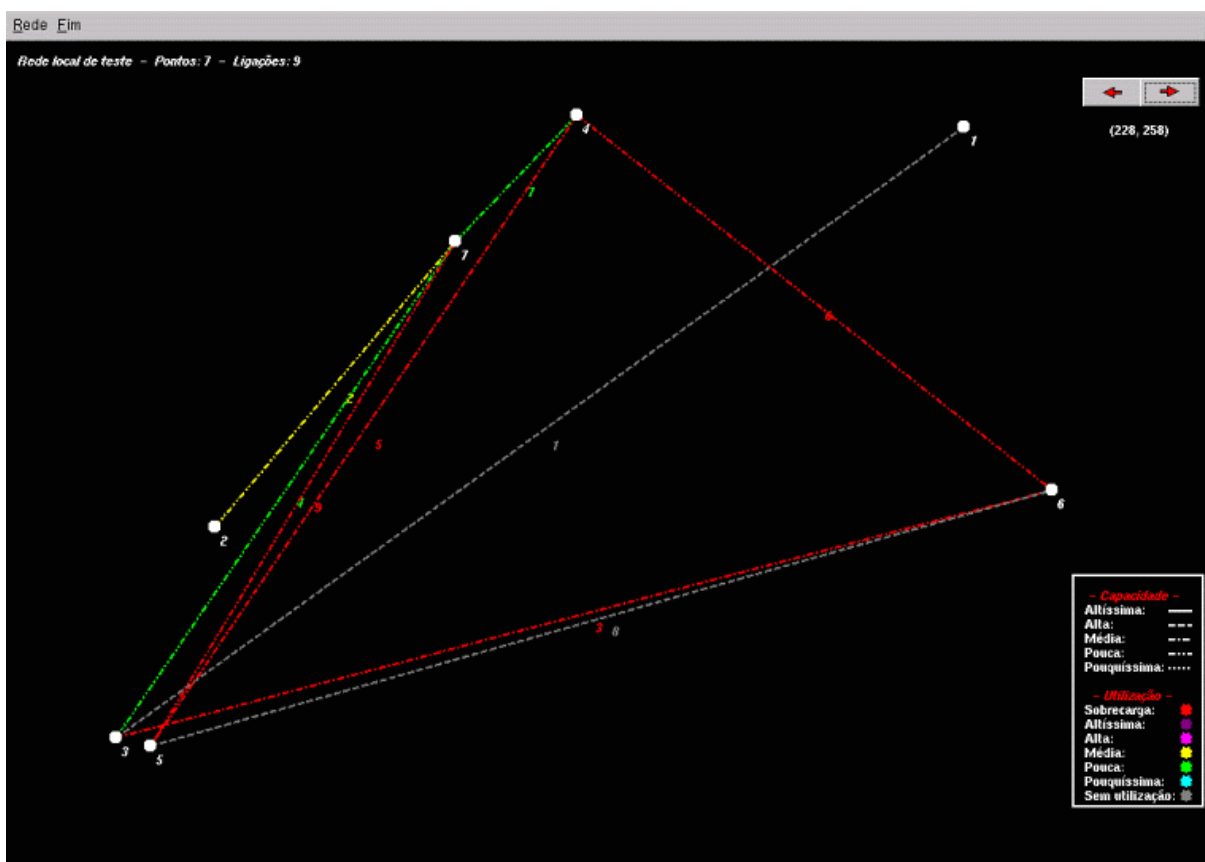


FIG. 4.10: Visualização da instância 7 de REDE1.

Foi apresentado um breve exemplo de como se pode ler a rede visualizada, extensível para o caso de outros pontos e ligações. A REDE1 foi gerada aleatoriamente, o que reforça o caráter genérico do KitNet, pois pode-se perceber sua utilização para a leitura e representação de sistemas e redes de serviço de qualquer espécie, conforme citamos anteriormente (seção 1.2).

Para tanto, supondo esta ser uma rede de comunicação de dados, até mesmo conforme consta em “Informações gerais”, o que pode significar, em momentos distintos, as alterações ocorridas na ligação (1, 2)? Na instância 6 é apresentada

com alta capacidade, na seguinte, deixa de existir, e na oitava, reaparece com média capacidade.

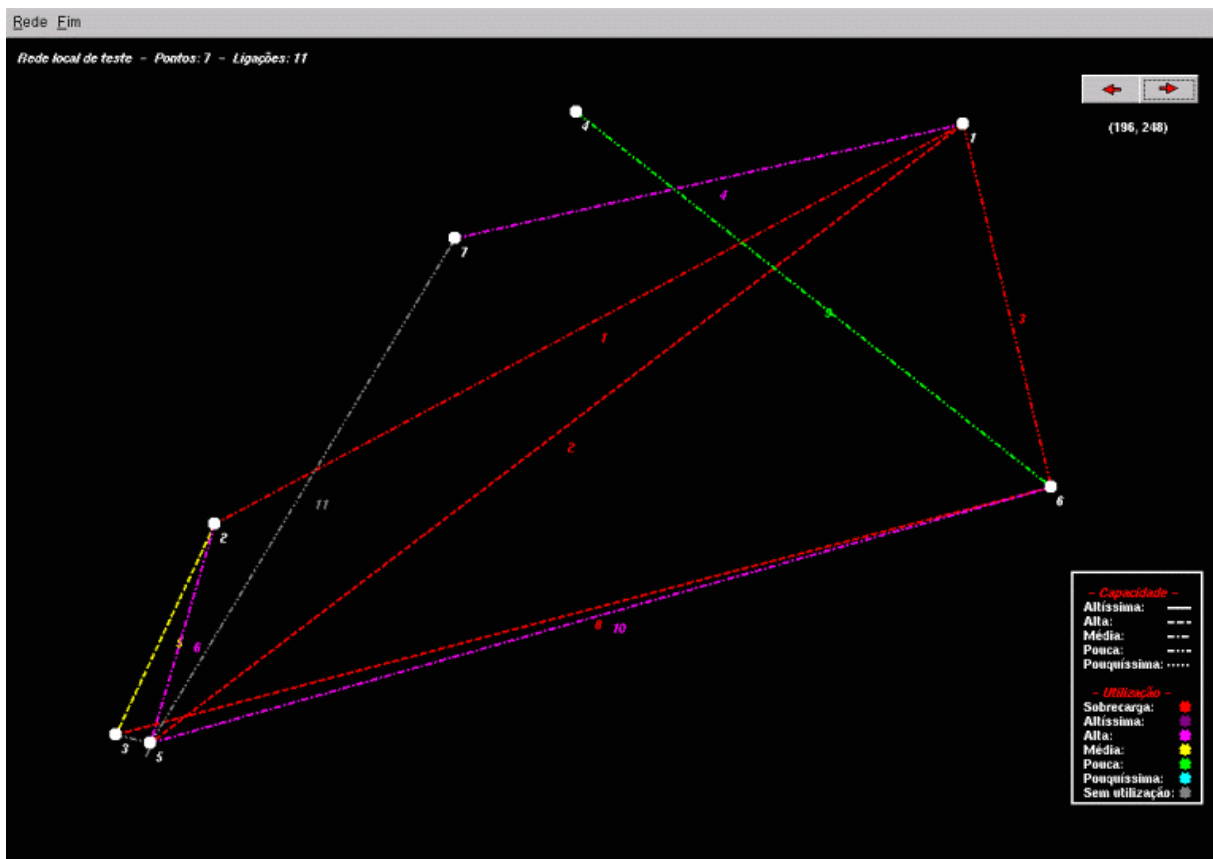


FIG. 4.11: Visualização da instância 8 de REDE1.

Por ser uma situação hipotética, pode-se supor a representação de um rede com links que permitem a reserva de largura de banda, em função do tipo de tráfego. Por exemplo, um mesmo link de comunicação compartilhado por uma transmissão de vídeo e outra de dados. Sob um protocolo voltado para qualidade de serviço (QoS), teríamos um percentual da capacidade reservada exclusivamente para a transmissão de vídeo, o que resultaria na alteração da capacidade nominal de transmissão.

Supondo agora que REDE1 represente um trecho de malha rodoviária as alterações apresentadas em (1, 2) podem representar uma via em pleno funcionamento, a mesma via totalmente interdita, por um acidente, por exemplo, e, momentos após, a via parcialmente liberada. Em ambos exemplos de situações reais, o KitNet, como outros software de visualização, se mostra uma

boa ferramenta para fins didáticos ou práticos, facilitando de forma efetiva o entendimento dos casos apresentados.

4.4 EDIÇÃO DA REDE

Com procedimento semelhante à criação da rede de forma interativa, podem ser feitas as operações para edição da rede visualizada, não importando como a mesma foi criada. Ou seja, para a edição de uma rede, basta que esta esteja sendo visualizada, enquanto que sua criação pode ter sido feita por qualquer modo, criada de forma automática, interativa ou obtida através de importação de seus dados.

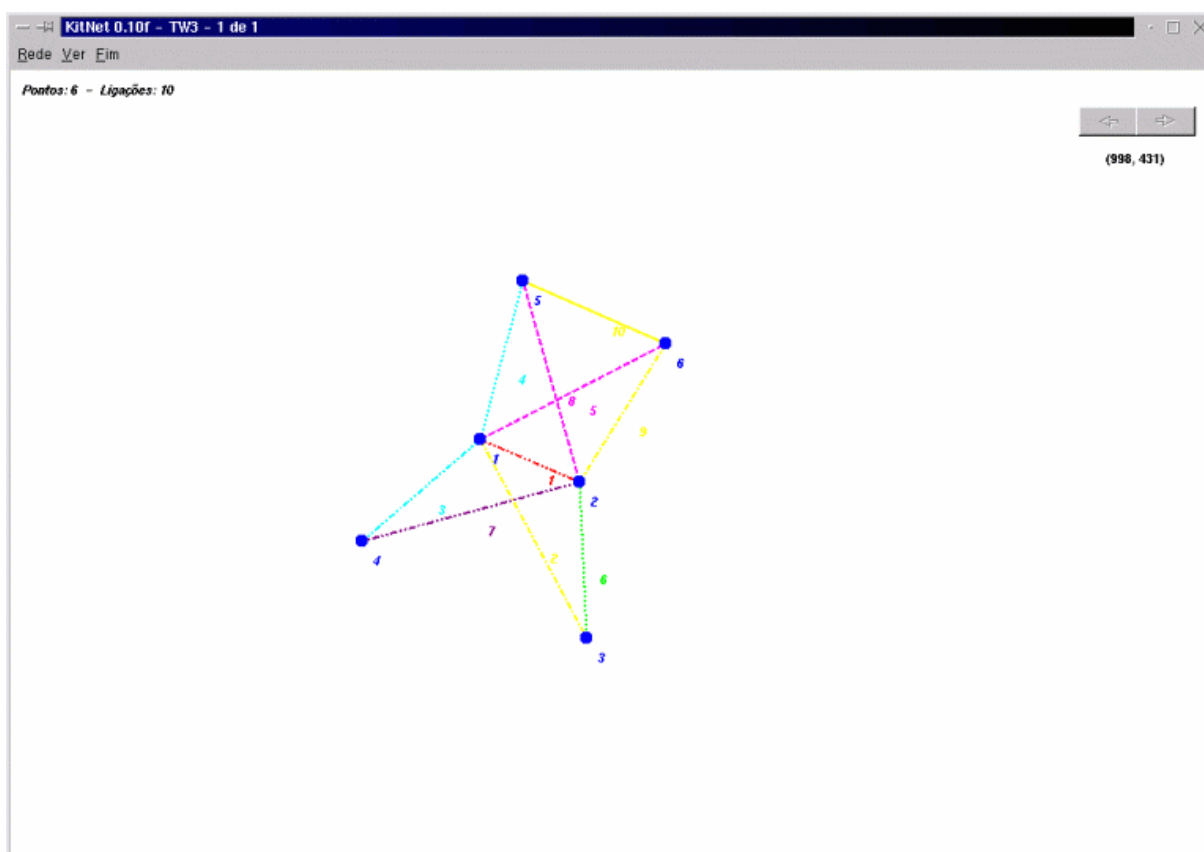


FIG. 4.12: Exemplo de rede a ser editada.

Uma vez visualizada pode-se, através do menu de procedimentos (FIG. 4.1), criar novos pontos ou ligações, alterar a posição de pontos e, por conseguinte, de suas ligações incidentes, e remover pontos e suas ligações incidentes ou

somente remover uma ligação. Sendo que qualquer alteração feita na rede pode ser gravada através do menu “Rede” (FIG. 4.2). Caso seja feita alteração em apenas uma instância da rede, então será refletida apenas na instância modificada.

A partir da FIG. 4.12, é apresentado um exemplo de edição em uma rede com 6 pontos e 10 ligações, com ligações de altíssima, alta, média, pouca e pouquíssima capacidade, que apresentam sobrecarga, altíssimo, alto, médio, pouco e pouquíssimo uso efetivo.

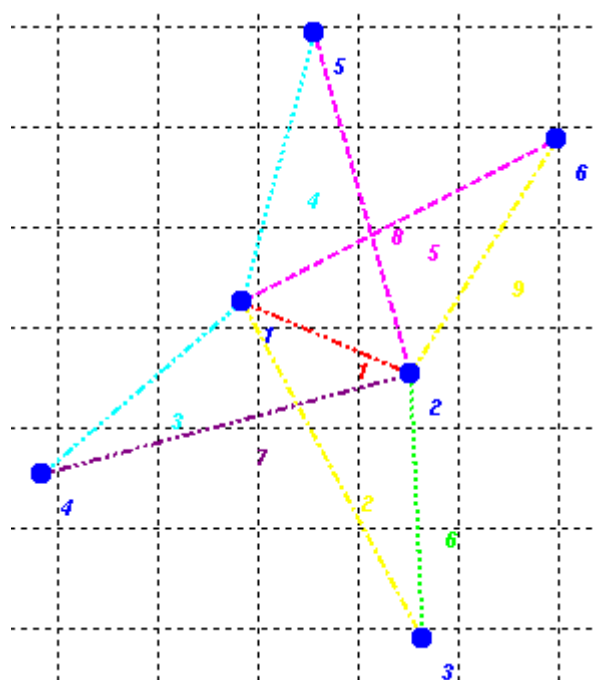


FIG. 4.13: Remoção da ligação 10.

As FIG. 4.13 a 4.19 apresentam momentos de edição da rede, onde se vê a remoção de uma ligação, de um ponto e conseqüentemente suas ligações incidentes, a movimentação de um ponto em dois tempos, apresentando a seleção do ponto e sua nova localização, a remoção de uma ligação, a remoção de um ponto, a inserção de um ponto e a inserção de uma ligação, também em dois tempos, com a seleção do ponto origem da ligação e a ligação estabelecida.

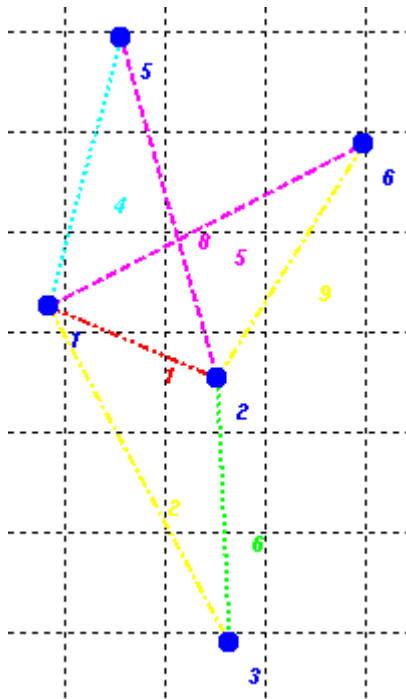


FIG. 4.14: Remoção do nó 4.

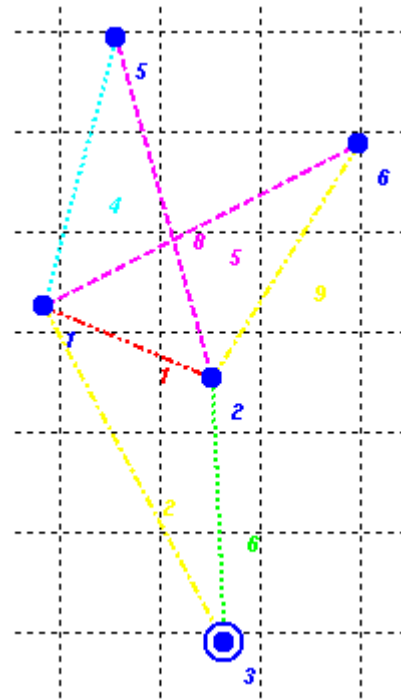


FIG. 4.15: Seleção do ponto 3 para ser reposicionado.

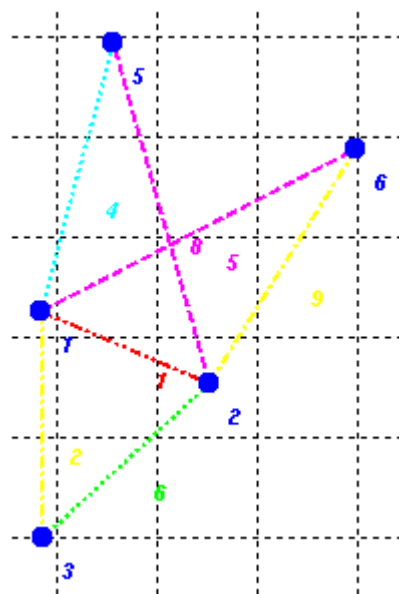


FIG. 4.16: Reposicionamento do ponto 3.

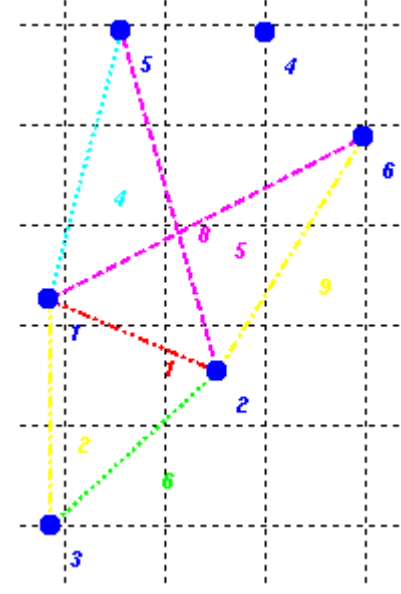


FIG. 4.17: Inserção de um novo ponto.

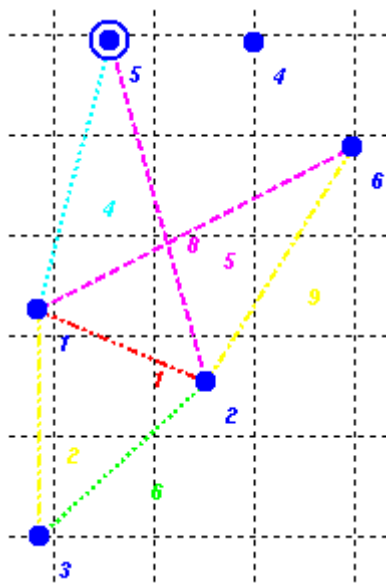


FIG. 4.18: Seleção do ponto 5 para ser origem da nova ligação.

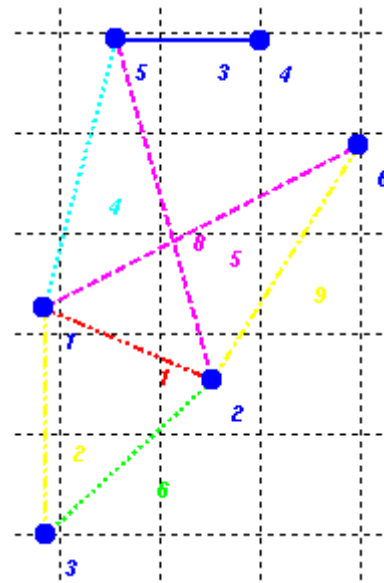


FIG. 4.19: Inserção da nova ligação, (5, 4).

4.5 VISUALIZAÇÃO

A visualização de uma rede no KitNet é implementada através da disponibilização de traçados automáticos para o usuário. Este é um dos diferenciais do projeto, visto que o levantamento e análise de trabalhos na literatura (seção 2.1 e 2.2) mostrou que os autores aos quais tivemos acesso desprezavam pouca preocupação com esse ponto. Entretanto, oferecer ao usuário a possibilidade de enxergar o sistema representado por diferentes desenhos, que podem representar diferentes topologias, fornece novas perspectivas de leitura da rede e facilita seu entendimento.

O protótipo disponibiliza através do menu de procedimentos (FIG. 4.1) quatro opções de traçados: o traçado aleatório, o traçado em coroa, o traçado segundo o algoritmo de Kamada e o traçado em níveis. Além disso, permite combinar os traçados com as buscas apresenta as redes sob novas óticas, onde o usuário pode conhecer os percursos e as ligações alternativas sobre cada um dos traçados. No anexo “A”, é apresentado um exemplo bastante completo das

possibilidades de visualização criadas com os traçados e as buscas sobre cada traçado.

O traçado aleatório é uma visão simples de novas possibilidades de desenho da rede, bastante útil para casos de testes. Os demais traçados buscam harmonizar o desenho para facilitar a leitura da rede. O traçado em coroa faz um distribuição equilibrada e totalmente simétrica dos pontos da rede, já o traçado pelo Kamada busca o equilíbrio em relação às ligações, ou melhor, em relação a distância entre os pontos, e o traçado em nível, particularmente útil em se tratando de redes, como parte de uma busca em largura, desenha a rede com o menor número de saltos entre a origem e os demais pontos da rede.

O traçado aleatório simplesmente sorteia os pontos da rede dentro da área de exibição, tomando o cuidado para não coincidir dois pontos em uma mesma posição, em seguida a rede é redesenhada seguindo o novo traçado e mantendo-se as propriedades correntes dos pontos e ligações.

Para a implementação do traçado em coroa tomou-se como base o programa “Kn”, cujo fonte encontra-se listado em MARKENZON & VERNET (1997), que calcula coordenadas polares⁶, X e Y , para os pontos e os distribui sobre um circunferência de raio R , com os respectivos valores sendo obtidos conforme mostrado a seguir. NP é o número de pontos da rede, H e L são, respectivamente, a altura e a largura da janela de exibição e C uma constante para a obtenção de um raio bem dimensionado para o tamanho da janela de exibição.

$$R = \frac{C \times NP}{2}; R \leq \frac{H}{2}$$
$$X = R \times \cos \phi + \frac{L}{2}; \frac{2\pi}{NP} \leq \phi \leq 2\pi$$
$$Y = R \times \sin \phi + \frac{H}{2}; \frac{2\pi}{NP} \leq \phi \leq 2\pi$$

Para ilustrar alguns traçados disponíveis para visualização, através do KitNet, foi gerada de forma interativa uma rede (FIG. 4.20), com 13 pontos e 18 ligações, a partir o grafo apresentado em SZWARCFITER (1986), página 105, e

⁶ Coordenadas na forma trigonométrica ou polar que descrevem o plano de Argand-Gauss (IEZZI et al., 1977).

n_1, \dots, n_n , l_{ij} é o comprimento desejado entre dois pontos, que é calculado a partir do caminho mais curto entre n_i e n_j , e k_{ij} é a força existente entre n_i e n_j , que é inversamente proporcional ao caminho mais curto entre os pontos. O objetivo do algoritmo, então, é minimizar E , através da resolução iterativa de um sistema de equações baseado no método de Newton-Raphson.

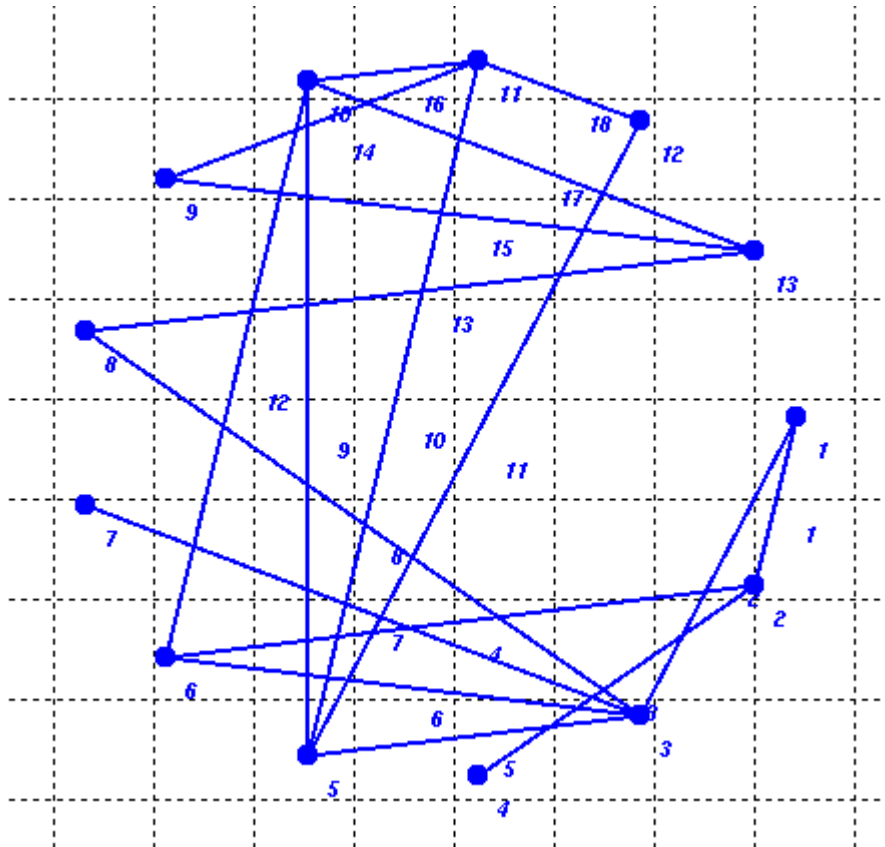


FIG. 4.21: Exemplo de SZWARCFITER (1986) traçado em coroa.

No desenvolvimento deste traçado foram detectadas algumas condições restritivas para a implementação deste algoritmo, por exemplo, o algoritmo não trata redes desconexas e é bastante comum não convergir para redes com um número maior de ligações ou cujo os pontos estejam de alguma forma alinhados. Além disso, QUARESMA & LOPES (1991) afirmam que as constantes utilizadas no algoritmo devem ser diferentes de grafo para grafo e de desenho para desenho, sob pena de o algoritmo não atingir uma posição estável, ou seja, não convergir e, assim como na implementação desta dissertação, foi necessário o desenvolvimento de um detector de ciclos para os casos em que o algoritmo não atingisse um ponto estável.

O desenho para a rede da FIG. 4.20 visualizado através do traçado pelo Kamada pode ser visto na FIG. 4.22.

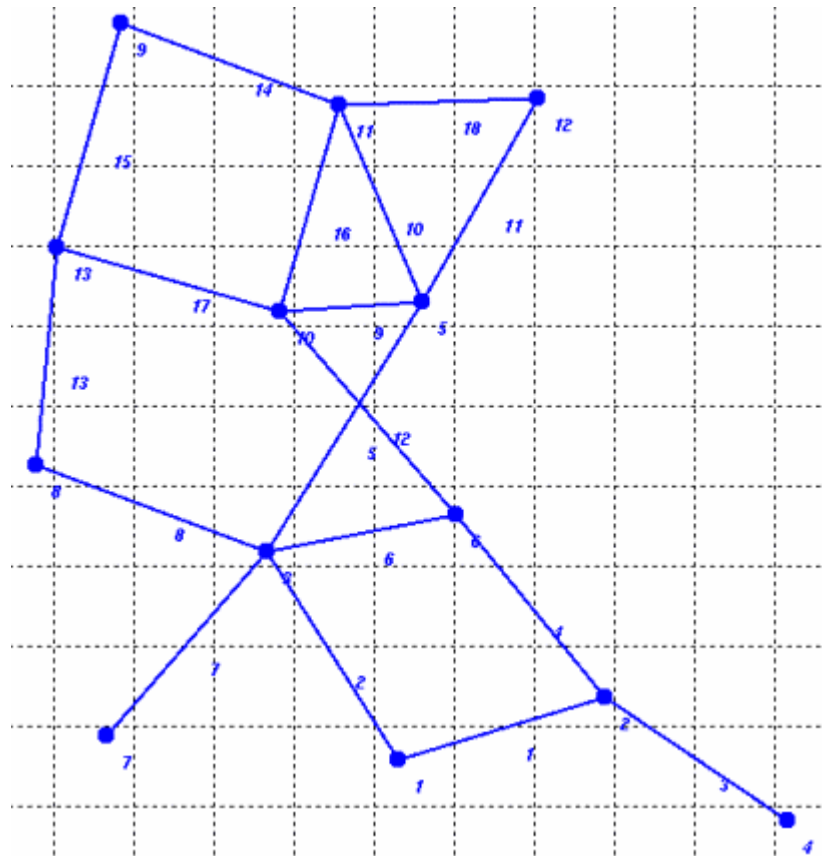


FIG. 4.22: Exemplo de SZWARCFITER (1986) traçado pelo Kamada

O quarto e último desenho de traçado disponibilizado ao usuário é o traçado em níveis, que cria e representa visualmente uma relação hierárquica na rede. Essa hierarquia é obtida por meio da árvore de caminhos gerada por busca em largura, iniciada em um ponto (raiz) origem da busca, que deve ser selecionado através do menu de procedimentos (FIG. 4.1), nas opções “Selecionar Ponto” ou “Localizar Ponto...”.

A busca em largura foi implementada segundo o algoritmo proposto por SZWARCFITER (1986), onde, durante a busca, é determinada a ordem dos pontos visitados, dada pela largura dos pontos e, como consequência, são obtidas as ligações do caminho⁷ percorrido, as arestas pai. As ligações que não fazem parte do caminho, foram chamadas de ligações alternativas, sem a

⁷

Ligações de caminho: —

Ligações alternativas: - - -

preocupação de separá-las em arestas tio, irmão ou primo, ao contrário da citada fonte.

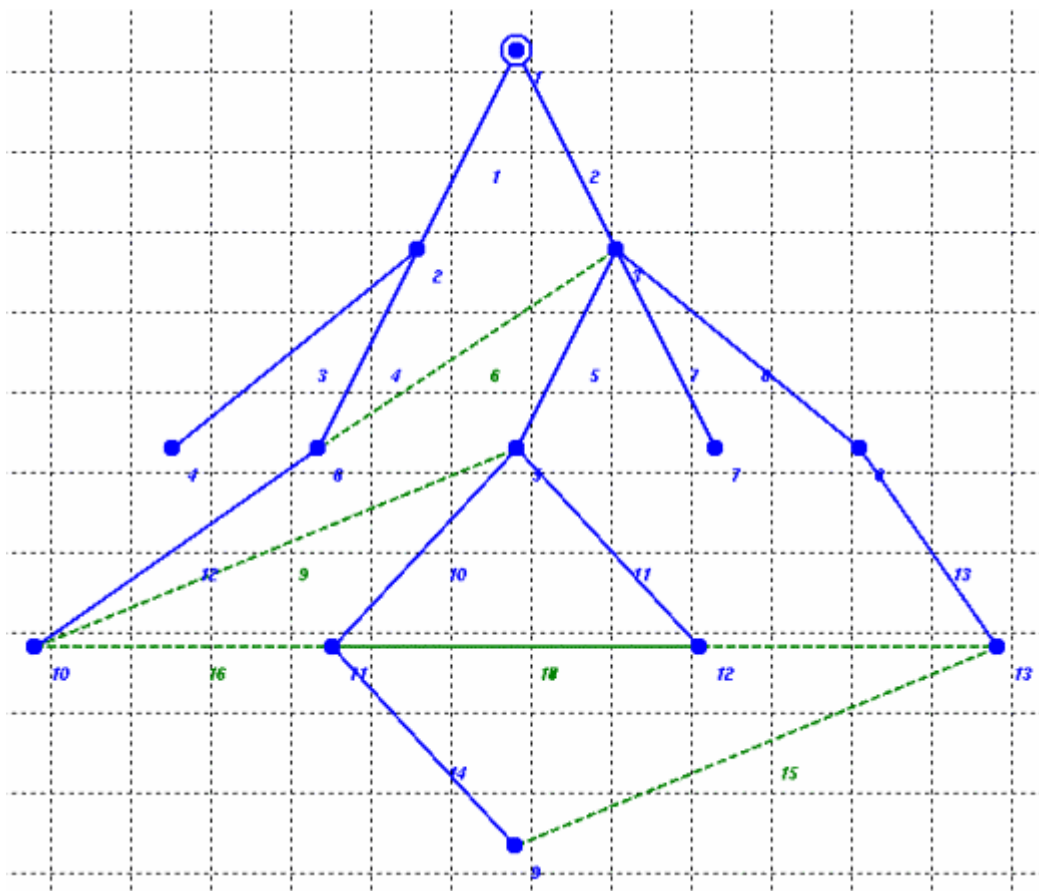


FIG. 4.23: Exemplo de SZWARCFITER (1986) traçado em níveis gerado por busca em largura a partir do ponto 1, com ligações alternativas visíveis e posicionamento reto dos pontos.

Neste algoritmo foram utilizados os recursos de criação e manipulação de atributos de elementos oferecidos pela GrBasic, sendo que cada ponto recebe dois atributos que serão utilizados no traçado em níveis: o atributo que informa o nível no qual o ponto se encontra e o atributo que informa qual a ordem em que o ponto deve ser desenhado no traçado, o que é diferente da largura do ponto, para que não haja cruzamento de ligações de caminho. O desenho para a rede baseada no exemplo de SZWARCFITER (1986) visualizado através do traçado em níveis pode ser visto na FIG. 4.23.

No protótipo KitNet, foi adotado como rótulo dos pontos e ligações o próprio identificador utilizado internamente nas estruturas da GrBasic, uma numeração seqüencial, inteira e crescente, e esta identificação é utilizada para determinar a ordem em que os pontos serão visitados, partindo de seu pai.

O ponto raiz, origem da busca, encontra-se no nível “1” e sua ordem para ser desenhado também será “1”. Em seguida os filhos do ponto raiz estarão no nível “2”, os filhos destes no nível “3” e assim sucessivamente. A determinação da ordem em que os pontos serão desenhados é tomada a partir do estabelecimento de valores ao respectivo atributo acordo com as seguintes regras:

A) A linha imaginária, ou raio imaginário, de posicionamento dos pontos de um mesmo nível deve ter tantos locais de posicionamento quantos forem os pontos que existem no nível, independente de possuírem, ou não, pais diferentes. Sendo que cada ponto ocupará um dos locais determinados.

Por exemplo, na FIG. 4.23 o nível “3” possui ao todo 5 pontos, filhos dos pontos “2” e “3”, assim a linha imaginária foi calculada com espaçamento para 5 pontos;

B) Se o ponto i encontra-se a esquerda do ponto j , então os filhos de i devem estar todos a esquerda dos filhos de j .

Por exemplo, também na FIG. 4.23, os filhos do ponto “2” são os pontos “4” e “6”, e os filhos do ponto “3” são os pontos “5”, “7” e “8”. Assim, os pontos “5”, “7” e “8” se encontram à direita de “4” e “6”.

C) Se i é menor do que j e se i e j são filhos do mesmo pai, então i deve ficar a esquerda de j .

Por exemplo, nesta mesma FIG., temos os pontos “5”, “7” e “8”, filhos do ponto “3”, em ordem crescente da esquerda para a direita.

Como a busca em largura implementada distingue as ligações em ligações de caminho e alternativas, o protótipo permite visualizar os traçados, qualquer um deles, com ou sem as ligações alternativas.

Para tanto, basta executar a busca, através do menu de procedimentos (FIG. 4.1), marcar ou desmarcar a opção “Ligações Alternativas” no menu “Ver” (FIG. 4.4), e em seguida solicitar o procedimento de traçado desejado, também através do menu de procedimentos.

Como exemplo, na FIG. 4.23, apresenta o traçado em níveis com as ligações alternativas visíveis, já a FIG. 4.24 apresenta o mesmo traçado, porém as ligações alternativas não aparecem. Essa possibilidade existe para qualquer traçado, tanto para busca em largura, quanto para busca em profundidade. No anexo “A” existem alguns exemplos de visualização nos diversos traçados com as duas buscas, e com ligações alternativas visíveis ou não.

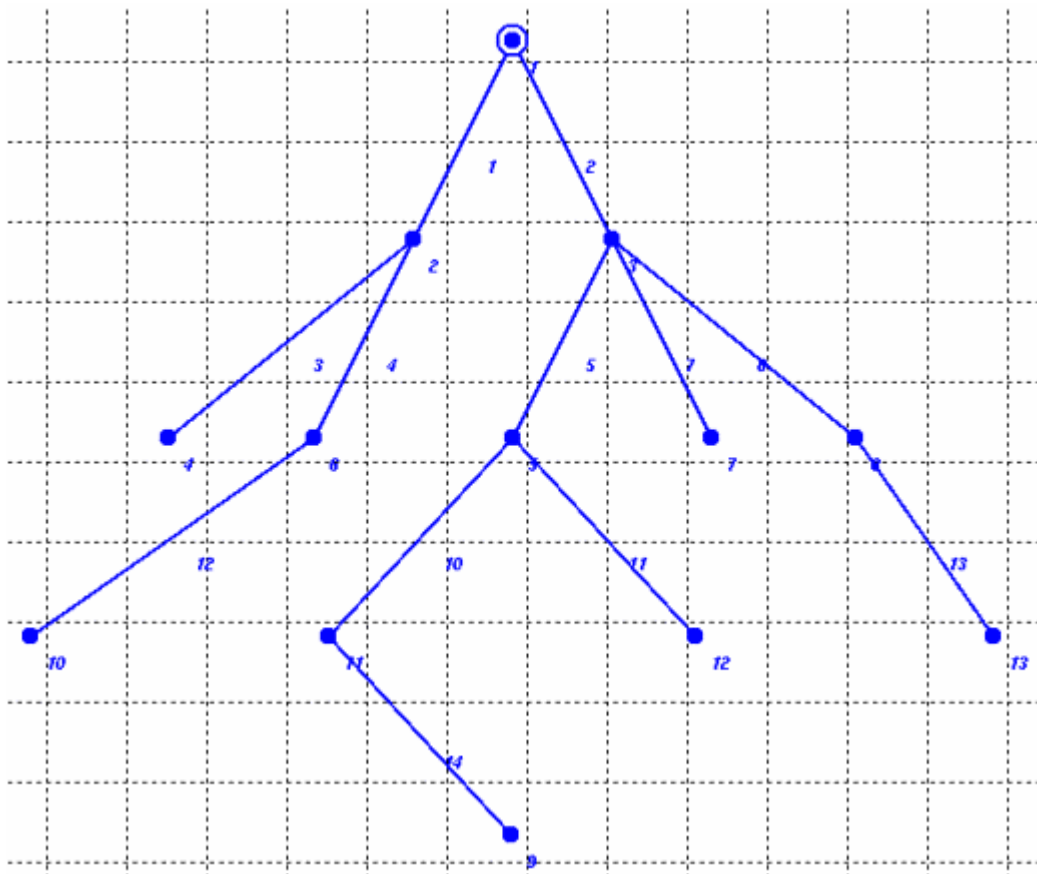


FIG. 4.24: Exemplo de SZWARCFITER (1986) traçado em níveis gerado por busca em largura a partir do ponto 1, com ligações alternativas não visíveis e posicionamento reto dos pontos.

Para o traçado em níveis, através do menu “Ver” (FIG. 4.4), são disponibilizados dois desenhos, com posicionamento reto dos pontos, opção padrão, ou com posicionamento radial dos pontos. No primeiro caso, os pontos são distribuídos, em cada nível, sobre uma linha imaginária fixada a uma distância Y_i da raiz e mantendo uma distância DN do nível anterior. Calculado como visto a seguir, considerando H a altura da janela de exibição, n o maior nível da árvore de busca, i o nível corrente e C uma constante utilizada para obter uma distância bem dimensionada para o tamanho da janela de exibição.

$$DN = \frac{H}{n}$$

$$Y_i = DN \times (i-1) + C$$

Habilitando a opção de posicionamento radial, com pode ser visto o exemplo na FIG. 4.25, os pontos são distribuídos em semicircunferências concêntricas com centro no ponto raiz, sendo uma semicircunferência para cada nível com raio Y_i , calculado como visto a seguir, mantendo-se as definições das variáveis e constantes:

$$Y_i = |\text{Sen}\phi| \times DN \times (i-1) + C; 0 \leq \phi \leq \pi$$

Em qualquer uma das opções de posicionamento a coordenada X é obtida por:

$$X_i = \text{Cos}\phi \times DN \times (i-1) + \frac{L}{2}; 0 \leq \phi \leq \pi$$

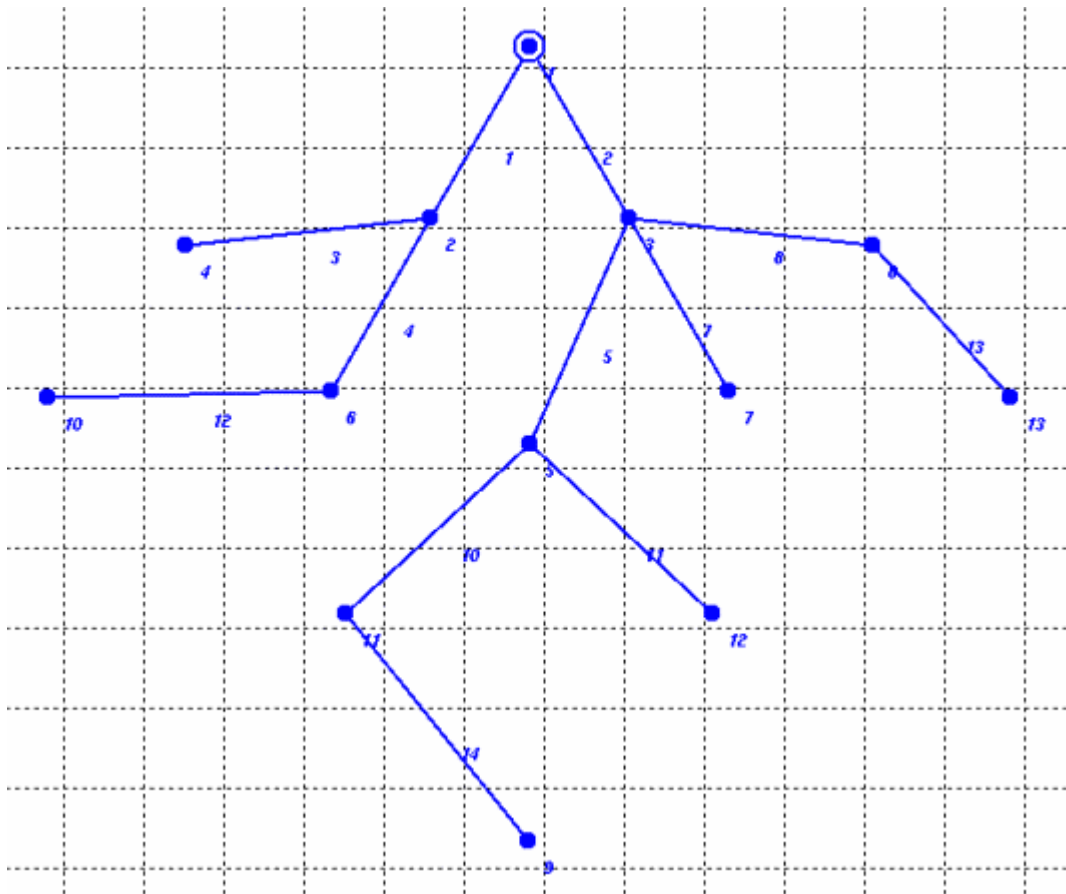


FIG. 4.25: Exemplo de SZWARCFITER (1986) traçado em níveis gerado por busca em largura a partir do ponto 1, com ligações alternativas não visíveis e posicionamento radial dos pontos.

Também mantendo-se todas as definições anteriores, L representa a largura da janela de exibição. No posicionamento radial as coordenadas X_i e Y_i também

são polares e geram o plano de Argand-Gauss, da mesma forma como no traçado em coroa.

As buscas em largura, já apresentada na seção 4.5, e em profundidade foram implementadas sem o uso do KineGraph e seguindo, respectivamente, o algoritmo de SZWARCFITER (1986) e o programa “*Busca_em_Profundidade*”, cujo o fonte encontra-se em MARKENZON & VERNET (1997).

4.6 DESENVOLVIMENTO

Nesta seção serão descritas as demais características, propriedades e funcionalidades do protótipo, que não foram abordadas até o presente momento, a partir dos procedimentos e opções de tratamento e manipulação, disponíveis nos menus “Rede”, “Ver” e de procedimentos (FIG. 4.1, 4.2, 4.3, 4.4), utilizados sobre as estruturas de armazenamento já citadas na seção 4.2, o arquivo de armazenamento da estrutura de dados da rede em meio não volátil, e a estrutura de dados da rede para manipulação interativa.

4.6.1 LEITURA, GRAVAÇÃO E DESCRIÇÃO DAS ESTRUTURAS DE DADOS

A opção “Criar...”, menu “Rede” (FIG. 4.2), já foi descrita na seção 4.3 em termos de funcionamento, cabendo apenas descrever a estrutura de dados para manipulação interativa gerada e que recebe o nome de *Rede*.

A estrutura *Rede* guarda todos os dados necessários a exibição e desenho da rede: o evento ou procedimento selecionado (inserção de ponto ou ligação, seleção de ponto, remoção de ponto ou ligação e movimentação de ponto) para atuar sobre a instância apresentada, controle da exibição ou não das informações de contexto (grade, moldura, informações gerais, legenda e coordenadas), controle das cores de exibição (cor do vértice, do fundo, da fonte, da grade e da moldura), variáveis utilizadas nos procedimentos (ponto e ligação correntes, pontos origem e destino de uma ligação), o descritor do arquivo que armazena ou armazenará a rede em disco, o nome base dos arquivos de referência e das instâncias, o título da rede que, se existir, comporá o conjunto de “Informações Gerais”, o controle das instâncias (o número de instâncias, a instância corrente

em exibição e ponteiros para as estruturas que armazenam a primeira instância, a última e a instância corrente) e, por fim, a estrutura que armazena o grafo corrente em exibição definida na GrBasic.

Além do procedimento de criação, as opções “Visualizar...”, “Gravar...”, “Importar...” e “Exportar...”, todas do menu “Rede” (FIG. 4.2), e manipulam diretamente a estrutura de dados *Rede* e, respectivamente, possibilitam carregar do disco um arquivo de armazenamento no formato KitNet para a estrutura *Rede*, gravar o conteúdo de *Rede* em um arquivo no formato KitNet, carregar para *Rede* arquivos em formato texto padronizado ou no formato KineGraph e, ainda gravar o conteúdo de *Rede* em formato texto padronizado.

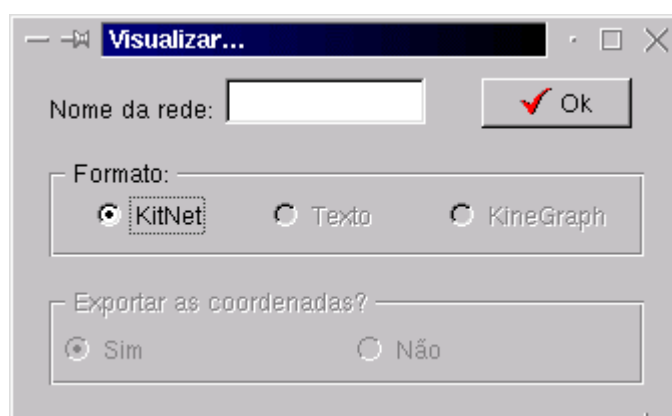


FIG. 4.26: Tela “Visualizar...”

Essas opções partem de uma tela que multifuncional que, dependendo da opção selecionada, disponibiliza os elementos referentes a ação desejada. Por exemplo, a tela a vista na FIG. 4.26 surge quando escolhe-se “Visualizar...” e, para tanto disponibiliza somente o campo para leitura do nome da rede. Os demais elementos aparecem indisponíveis ou já selecionadas, sem que o usuário possa alterar. Além disso, dependendo da rotina chamada, o título da tela, no caso se vê “Visualizar...”, é alterado para ajudar na identificação da solicitação.

Outros elementos, tais como, selecionar formato texto ou KineGraph, ou permitir a exportação das coordenadas somente estarão disponíveis quando for conveniente. Por exemplo, opção de exportação de coordenadas somente estará acessível em “Exportar...”.

Considerando que foi acionada a rotina de visualização e fornecido o nome da rede solicitado, o “nome base”, o KitNet abrirá para leitura os dois arquivos que armazenam a rede em disco, o arquivo de referência e controle (“.knt”) e os

arquivos das instâncias (".gra"), este último definido na GrBasic, quando então os dados armazenados em disco serão carregados para a estrutura *Rede* de manipulação interativa.

O arquivo ".knt" é gravado em formato texto e possui somente três registros: o primeiro com o nome base da rede, o segundo com o título da rede, utilizado na exibição, se este existir, e caso não exista, no segundo registro armazena-se a cadeia "*Sem Título*". O terceiro e último registro do arquivo ".knt" guarda o número de instâncias da rede, ou seja, também informa quantos arquivos ".gra" existem para a rede em questão. O nome de todos os arquivos são compostos a partir do nome base.

Por exemplo, supondo uma rede com nome base "REDE1" e quatro instâncias, existirá o arquivo "REDE1.knt", que será o arquivo de referência, e os arquivos "REDE1_1.gra", "REDE1_2.gra", "REDE1_3.gra" e "REDE1_4.gra", que armazenarão as instâncias da rede.

A rotina de gravação executa um procedimento inverso à de visualização, pois parte da estrutura *Rede*, já citada anteriormente, e cria a estrutura de armazenamento da rede em meio não volátil, ou seja, os arquivos de referência e das instâncias, no formato e conteúdos já descritos.

4.6.2 IMPORTAÇÃO E EXPORTAÇÃO DE DADOS

A rotina de importação permite ler arquivos no formato original do KineGraph ou arquivos textos em formato determinado, conforme serão descritos logo a seguir.

Os arquivos do KineGraph são arquivos ".gra" e, uma vez importados, são transformados em arquivos do sistema operacional GNU/Linux e associados a um arquivo ".knt" de referência da rede que descreve.

Os arquivos textos para importação devem atender a uma pequena especificação de formato, que será explicado a partir do exemplo mostrado na FIG. 4.27. O exemplo está com as linhas numeradas apenas para ser usado

como referência no decorrer deste texto, sendo o arquivo que originou a rede exibida no exemplo “C”, vista no anexo “A”.

```
1. 35
2. 1 46 78 23
3. 4 c 67 89 23 56 89
4. 767 3
5. 43 r TESTE
6. 6 10 20 30 40 50 60 70 80 90 100 11 22 33 44 55 66 77 88 99
7. 8 c 67 89 23 56 89
8. 19 *
9.
10. 6767 89 23 56 89
11. 34 67 89 23 56 89
12. 90
13. 2 67 89 23 56 89
14. 45
15. 67 67 89 23 56 89
16. 34
17. 82 67 89 23 56 89
18. 83
19. 84 67 89 23 56 89
20. 85
21. 11 67 89 23 56 89
22. 15
23. 17 67 89 23 56 89
24. 18
25. 21 67 89 23 56 89
26. 22
27. 35 67 89 23 56 89
28. 38
29. 47 67 89 23 56 89
30. 56
31. 53 67 89 23 56 89
32. 62
33. 68 67 89 23 56 89
34. 74
35. 77 67 89 23 56 89
36. 67 89 23 56 89
37.
38. 89 23 56 89
39. 23 56 89
40. 56 89
41. 898
```

FIG. 4.27: Exemplo de arquivo texto para importação.

Um arquivo texto para importação parte dos seguintes princípios:

- A) A primeira linha deve conter somente a quantidade de pontos da rede. Assim, o protótipo ignora qualquer caracter até que surja um algarismo e, a partir deste, identifica o número iniciado por este algarismo, descartando o restante dos caracteres que, por ventura, existam na linha.

- B) O primeiro número, o primeiro campo, de cada linha identifica o ponto cuja informações constam nesta mesma linha, sendo, no momento, o ponto corrente.
- C) Após a identificação do ponto corrente, todas as demais informações serão outros pontos que estarão ligados ao ponto corrente ou coordenadas de localização do ponto corrente. Se não existir a informação das coordenadas, a localização do ponto corrente será aleatória.
- D) As coordenadas são identificadas pelos caracteres “c” ou “C” seguidos de dois valores, respectivamente, coordenadas X e Y. Se algum dos dois, ou os dois, não existir ou se estiver fora dos limites da janela de exibição, então o valor será escolhido aleatoriamente.
- E) Se existir na linha um asterisco, então indicará que o ponto corrente possui ligação para todos os pontos da rede.
- F) Os campos em uma linha serão separados por qualquer caractere diferente de algarismo, considerados simples separadores de campo e, conseqüentemente, ignorados.
- G) Um mesmo campo pode ter suas informações divididas em duas ou mais linhas, bastando que as linhas iniciem com a sua identificação.
- H) Informações duplicadas são ignoradas, mas informações redundantes porém lógicas, substituem a anterior. Por exemplo, se forem informadas duas ou mais coordenadas para um mesmo ponto, será mantida a informação mais recente.

Assim, de acordo as regras especificadas as seguintes informações são obtidas da FIG. 4.27:

- 1) A linha “1” informa que a rede possui 35 pontos;
- 2) A linha “2” informa que existe a ligação (1, 23), e as demais informações serão ignoradas pois tratam de pontos que extrapolam o total de 35 pontos;

- 3) A linha “3” informa que o ponto “4” está localizado na posição (67, 89), que existe a ligação (4,23) e as demais informações serão ignoradas pela mesma razão do item anterior;
- 4) As linhas “4”, “5”, “10”, “12”, “14”, “15”, “17” a “20”, “28” a “36”, “38”, “40” e “41” serão ignoradas, pois os respectivos pontos correntes não existem em uma rede de 35 pontos;
- 5) A linha “6” informa que existem as ligações (6, 10), (6, 20), (6, 30), (6, 11), (6, 22) e (6, 33), e as demais informações serão ignoradas pela mesma razão do item 2;
- 6) A linha “7” indica que o ponto 8 se deve ser localizado nas coordenadas (67, 89), porém isto não será feito, pois esta mesma coordenada está ocupada pelo ponto “6”, então o ponto 8 ocupará uma posição aleatória. Informa também que existe a ligação (6, 23) e as demais informações serão ignoradas pela mesma razão do item 2;
- 7) A linha “8” informa que o ponto “19” está ligado com todos os demais 34 pontos;
- 8) As linhas “9” e “37” serão ignoradas por estarem em branco;
- 9) As linhas “11”, “13”, “21”, “23”, “25” e “27”, respectivamente, informam que existem as ligações (34, 23), (2, 23), (11, 23), (17, 23), (21, 23) e (35, 23) as demais informações serão ignoradas pela mesma razão do item 2;
- 10) As linhas “16”, “22”, “24” e “26” serão ignoradas pois não trazem nenhuma informação sobre os respectivos pontos correntes.
- 11) A linha “39” não terá efeito pois não traz nenhuma informação válida sobre o ponto corrente.

A rotina de exportação faz o trabalho inverso gerando arquivos no formato texto já descrito. Com a opção do arquivo ser composto somente com número de pontos da rede e as respectivas ligações ou, ainda, com as coordenadas dos pontos, bastando selecionar “Sim” em “Exportar as coordenadas?” na tela “Exportar...” (FIG. 4.28).

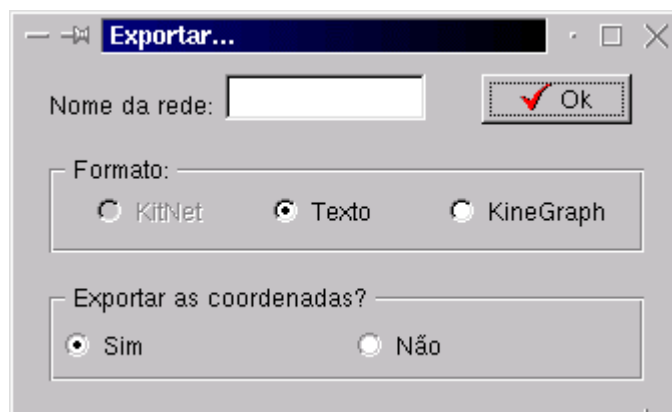


FIG. 4.28: Tela “Exportar...”

Esses dois casos podem ser vistos, respectivamente, nas FIG. 4.29 e 4.30, onde foi exportada rede do exemplo da FIG. 4.20.

```

13
1 2 3
2 1 4 6
3 1 5 6 7 8
4 2
5 3 10 11 12
6 2 3 10
7 3
8 3 13
9 11 13
10 5 6 11 13
11 5 9 10 12
12 5 11
13 8 9 10

```

FIG. 4.29: Arquivo de exportação do exemplo de SZWARCFITER (1986), sem as coordenadas dos pontos

```

13
1 2 3 c 324 300
2 1 4 6 c 456 189
3 1 5 6 7 8 c 456 428
4 2 c 340 178
5 3 10 11 12 c 553 436
6 2 3 10 c 556 202
7 3 c 303 428
8 3 13 c 595 284
9 11 13 c 642 322
10 5 6 11 13 c 856 323
11 5 9 10 12 c 706 338
12 5 11 c 578 339
13 8 9 10 c 656 289

```

FIG. 4.30: Arquivo de exportação do exemplo de SZWARCFITER (1986), com as coordenadas dos pontos

4.6.3 ESPECIFICAÇÃO DE OUTROS DETALHES RELEVANTES

O menu “Ver” (FIG. 4.3 e 4.4) trata das opções, características e propriedades, que permite ao usuário tornar a interface mais agradável e adequado ao uso, e que seus elementos já foram apresentados e discutidos nas seções 4.2. e 4.5. Cabe ressaltar poucos detalhes, tais como, a dimensão da área da janela de exibição delimitada pela moldura, que possui 881 pixels de largura, por 644 pixels de altura, perfazendo a possibilidade 567.364 diferentes localizações para os pontos das redes. Assim como a moldura, a grade auxilia

muito no desenho da rede, principalmente durante uma edição interativa, e esta encontra-se espaçada de 50 em 50 pixels.

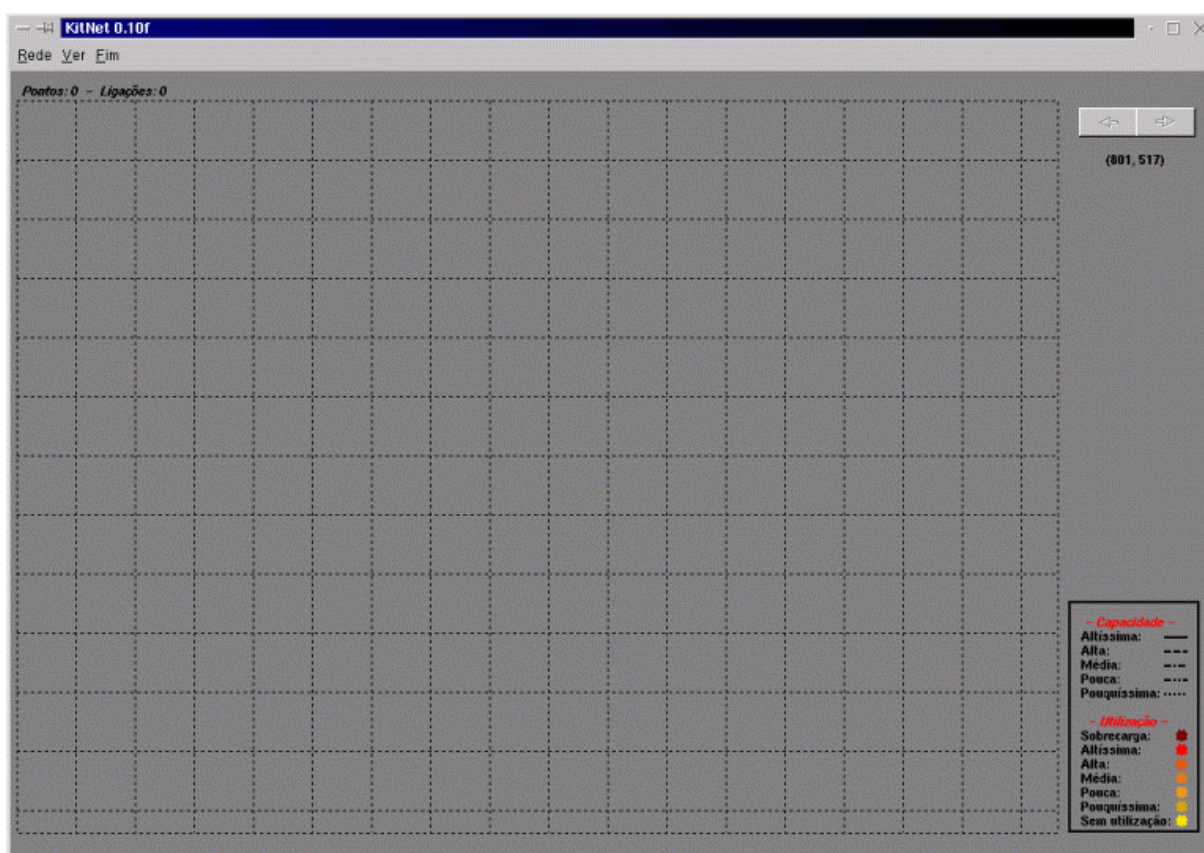


FIG. 4.32: Tela inicial do KitNet com cor de fundo cinza.

Outra facilidade que o protótipo oferece é a alteração da cor de fundo e, conseqüentemente, de alguns componentes visuais da tela. O KitNet oferece para cor de fundo as opções preto, como opção padrão, e ainda branco ou cinza. Exemplos com a tela utilizando a cor de fundo padrão podem ser vistos nas FIG. 4.5, A.1, A.2 e A.3. Com a opção da cor branca, utilizada em várias FIG., podem ser vistos nas FIG. 4.12 e C.1, e com a cor de fundo cinza pode ser visto na FIG. 4.31.

A possibilidade de alteração da cor de fundo é especialmente atrativa em situações de pouca luminosidade, fornecendo maior nitidez ou clareza, ou para oferecer maior conforto visual ao usuário.

O menu de procedimentos (FIG. 4.1) já foi abordado no decorrer das seções 4.2, 4.3, 4.4 e 4.5, contudo resta descrever detalhes dos procedimentos relacionados à edição, são eles, inserção de ponto e ligação, seleção e

localização de ponto, movimentação de ponto e remoção de ponto e ligação, já exemplificados na seção 4.4 (FIG. 4.12 a 4.19).

Para a inclusão de pontos, após a seleção do evento “Ponto”, basta que seja selecionado através do botão esquerdo do mouse uma localização desocupada dentro da janela de exibição, onde será posicionado o novo ponto (FIG. 4.17). A inclusão do ponto somente não terá sucesso, caso o local selecionado esteja ocupado por outro ponto ou não se encontrar dentro dos limites da janela de exibição.

A inclusão de uma nova ligação ocorre em dois passos. Após a seleção da opção “Ligação”, primeiramente deve ser selecionado o ponto origem da ligação e em seguida o ponto destino (FIG. 4.18 e 4.19). Apesar de não tratar de forma visualmente adequada, o KitNet permite que o ponto origem e destino seja o mesmo, criando um laço, assim como permite multi-ligações entre dois pontos.

O evento de movimentação de um ponto ocorre de forma similar, em dois passos (FIG. 4.15 e 4.16). Após escolhida a opção “Mover”, primeiro deve-se selecionar o ponto que será movido e, em seguida, a posição destino para o ponto, mantendo o cuidado de não selecionar uma posição destino que já esteja ocupada ou que se encontre fora da janela de exibição, pois, se assim for, a movimentação não será efetuada. Como consequência do reposicionamento de um ponto, todas ligações incidentes ao ponto serão movidas com o mesmo.

Selecionar um ponto significa marcar um determinado ponto para ser origem de um busca, de um traçado, ou dos eventos de inclusão de ligação ou de movimentação de ponto. Além disso, se por algum motivo uma operação de inclusão de ligação ou de reposicionamento de ponto foi interrompida após o primeiro passo mantendo um ponto origem marcado, esse ponto deve ser considerado como se estivesse sido selecionado através dos procedimentos “Selecionar Ponto” ou “Localizar Ponto...”

Para marcar um ponto através da opção “Selecionar Ponto”, basta simplesmente, após escolhida a opção, selecionar o ponto com o botão esquerdo do mouse. Porém em casos de redes de maior tamanho, fica difícil, visualmente, achar o ponto que se deseja selecionar. Assim, pode-se optar por “Localizar Ponto...” (FIG. 4.33), quando será apresentado um lista com o identificadores

(rótulos) de todos os pontos facilitando a seleção do ponto procurado. Nos dois casos, o ponto será marcado e sinalizado como nas FIG. 4.15 e 4.18.

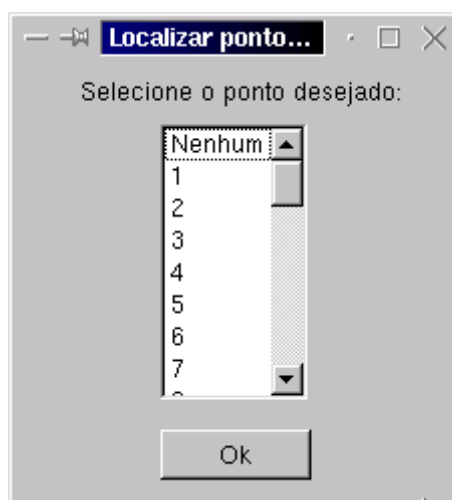


FIG. 4.33: Tela “Localizar Ponto...”

A remoção de um ponto ocorre simplesmente quando, após escolhido procedimento “Apagar”, seleciona-se o ponto com botão esquerdo do mouse e, como consequência, todas as arestas incidentes ao ponto serão removidas.

A remoção de uma ligação segue o mesmo princípio, entretanto existe uma dificuldade maior em identificar qual a ligação que está sendo selecionada para ser removida. Enquanto um ponto ocupa uma área⁸ desenhada de, aproximadamente, $78,5 \text{ pixel}^2$, o que facilita a seleção com o cursor do mouse, deve ser criado um recurso que facilite a seleção de um ligação, pois a mesma é representada por um segmento de reta com largura de no máximo 2 pixels, o que dificulta o posicionamento do mouse sobre a ligação.

Assim, foi implementada a função *IsOnEdge*, apresentada na FIG. 4.34 com as linhas numeradas para facilitar a explicação, que baseada em semelhança de triângulos, verifica se o cursor do mouse encontra-se muito próximo da ligação, ou exatamente sobre a ligação.

A função *IsOnEdge* verifica se o ponto P , em (X, Y) , está muito próximo, ou sobre a ligação E , com origem em *Rede.Alpha*, localizado em $(X1, Y1)$, e destino em *Rede.Omega*, localizado em $(X2, Y2)$, conforme ilustrado na FIG. 4.35.

⁸ O elemento ponto é representado visualmente como um disco de raio igual a 5 pixels, o que fornece uma área desenhada superior a $78,5 \text{ pixel}^2$.

```

1. function IsOnEdge (E: Edge; X, Y: integer): boolean;
2. var
3.     X1, Y1,
4.     X2, Y2: integer;
5.     Alfa, Beta: extended;
6. begin
7.     Rede.Alpha := alpha (E);
8.     Rede.Omega := omega (E);
9.     get_position (Rede.Alpha, X1, Y1);
10.    get_position (Rede.Omega, X2, Y2);
11.    //
12.    // Retorna aos valores default para nao influenciar nas demais rotinas
13.    // de desenho, alteracao, selecao, etc.
14.    //
15.    Rede.Alpha := NOVERTEX;
16.    Rede.Omega := NOVERTEX;
17.    //
18.    // Verifica se o ponto "clicado" esta entre as coordenadas dos
19.    // vertices inicial e final da aresta.
20.    //
21.    if (((X <= X1) and (X >= X2)) or ((X <= X2) and (X >= X1))) and
22.        (((Y <= Y1) and (Y >= Y2)) or ((Y <= Y2) and (Y >= Y1))) then begin
23.        Alfa := ArcTan2 ((Y1 - Y2), (X1 - X2));
24.        Beta := ArcTan2 ((Y1 - Y), (X1 - X));
25.        //
26.        // Verifica se o arco Beta (angulo dado pela intersecao da
27.        // reta estabelecida pelo ponto clicado e o vertice inicial
28.        // da aresta, com o eixo das abcissas) e "similar" ao arco
29.        // Alfa (angulo formado entre a reta que suporta a aresta
30.        // e o eixo das abcissas).
31.        //
32.        if (Beta <= (Alfa + 0.05)) and (Beta >= (Alfa - 0.05)) then
33.            IsOnEdge := TRUE
34.        else
35.            IsOnEdge := FALSE;
36.    end
37.    else
38.        IsOnEdge := FALSE;
39. end;

```

FIG. 4.34: Função *IsOnEdge*.

Nas linhas 23 a 35 (FIG. 4.34) é verificado se P está muito próximo do segmento $\overline{\alpha\omega}$, calculando e comparando as tangentes de ϕ e β . Caso os arcos se encontrem com uma variação máxima de 0.05 radiano, então é considerado que a ligação (α, ω) foi selecionada. Neste caso o ponto P satisfaz a condição, ao contrário do ponto P'' . Porém, o ponto P' também satisfaz essa mesma condição, mas não satisfaz a condição testada nas linhas 21 e 22 (FIG. 4.34), cuja regra estabelece que o ponto selecionado deve ser interno ao retângulo imaginário, onde uma das diagonais é o segmento $\overline{\alpha\omega}$.

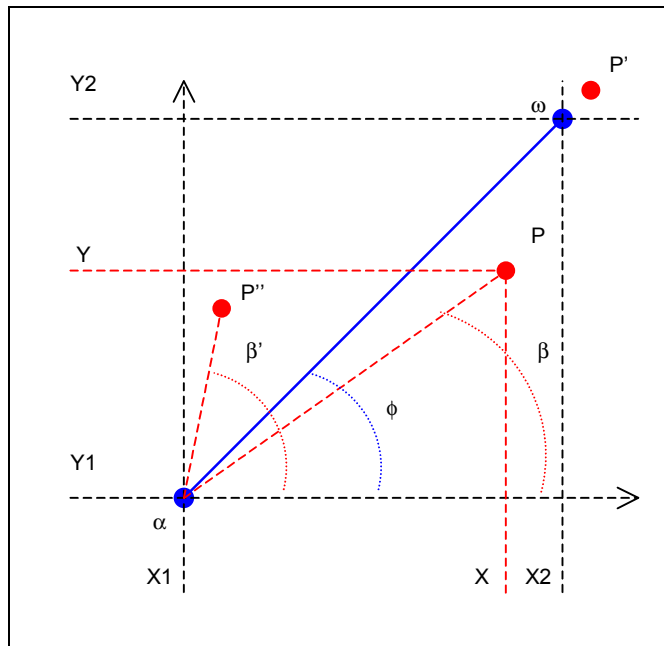


FIG. 4.35: Verifica se o ponto P seleciona a ligação (α, ω) .

4.7 COMENTÁRIOS E AVALIAÇÃO

O desenvolvimento do KitNet foi proposto, desde início, como um protótipo para validar a presente dissertação, não havendo a pretensão de desenvolver uma ferramenta completa. Entretanto, apesar de apresentar muitos pontos positivos, o protótipo possui outros tantos que mereciam especial atenção. Por exemplo, o fato de não tratar redes direcionadas especificamente, apesar de a própria estrutura das bibliotecas do KineGraph oferecer essa possibilidade.

Com relação aos aspectos visuais da interface, apesar do KitNet permitir multiligações entre os pontos a representação não é devidamente tratada, gerando, visualmente, uma representação confusa de um multigrafo. Além disso, deveria ter sido tratado com mais critério a disposição dos rótulos dos pontos e ligações, pois em alguns casos a leitura fica prejudicada.

O KitNet priva pela facilidade de operação e, assim sendo, disponibiliza a opção de criação da rede de forma aleatória e automática, com a intenção de permitir a visualização de um sistema mesmo que um usuário praticamente não saiba como criá-lo. Esta opção torna-se muito útil à medida que o usuário pode

utilizá-la como um ensaio e para aprender como manipular as características do software, além de ser um ótimo recurso para gerar redes para testes. Para tanto, o formulário desta opção traz valores predefinidos ou, então, sorteia ou infere os valores necessários para a criação de uma rede.

Ainda considerando procedimentos disponibilizadas e características do protótipo, foi identificada e defendida a necessidade do tratamento de bloco em SV, até mesmo por ser um diferencial em relação aos trabalhos apresentados, contudo, no KitNet este apresentou-se de forma incipiente, restringindo-se apenas a disponibilizar traçados para o maior bloco do sistema representado, ou seja, a rede como um todo.

A convenção adotada para a representação de valores, tais como para a representação de ligações com altíssima capacidade, alta capacidade, etc., apesar de bastante abrangente, poderia ser totalmente aberta permitindo a personalização da mesma por parte do usuário. Entretanto, foi mantida como se apresenta, pois, por ser intuitiva, facilita a qualificação, ou a classificação, sem a necessidade de quantificar ou estabelecer valores. Por exemplo, qualquer usuário, dentro do seu contexto, sabe o que significa “alta capacidade”, sem necessariamente ter que saber quanto significa em valor.

Não foi implementado, e realmente falta ao KitNet, uma simples operação de “desfazer”. Em um software com recursos de edição e editoração é uma opção extremamente útil. Da mesma forma, também nota-se a ausência da opção de exportação dos dados da rede ou grafo em todos os formatos que trata. Isto é, o protótipo possibilita a importação de grafos que não foram construídos no KitNet, a partir de arquivos em formato texto padronizado e em formato KineGraph. Entretanto, o software não oferece a possibilidade de exportar arquivo do KitNet para o formato KineGraph. Além disso, também falta ao KitNet um “Manual do Usuário”, que pode ser implementado como menu de ajuda ou texto.

Tecnicamente, sob o ponto de vista de desenvolvimento e implementação, analisando a plataforma, escolhida ressalta-se que a motivação para o uso do ambiente “Intel – GNU/Linux – Kylix” se dá, primeiramente, devido ao hardware ser uma plataforma muitíssima utilizada em qualquer ambiente computacional, em seguida, devido ao sistema operacional ser comprovadamente estável, de simples e fácil aquisição, possuir alta frequência de atualização e, ainda, ter a fatia de

mercado de utilização em franca expansão nas diversas áreas da computação. Dentre as inúmeras opções de distribuições existentes para o GNU/Linux, optamos pela Red Hat, por possuir o maior volume de uso entre as distribuições, principalmente se considerarmos as chamadas “Red Hat based”, além de disponibilizar suporte a um grande número de dispositivos de hardware.

Com relação ao ambiente de desenvolvimento, optou-se pelo Kylix, principalmente por, ao juntar-se com o Delphi, permitir a construção de um software multiplataforma, ou seja, podendo, o executável, ser gerado tanto para GNU/Linux, quanto para MS Windows.

Sob o aspecto de conceitos de linguagens e técnicas de desenvolvimento e programação, apesar de ter sido utilizada uma linguagem que tem com paradigma a orientação a objetos, a implementação do software não utilizou as devidas técnicas, apesar dos conceitos terem sido utilizados.

Com mais detalhes, é visível a relação de hierarquia e herança entre, por exemplo, “rede – grafo – vértice”, o que caberia a implementação de uma classe e classes derivadas, cada qual com suas propriedades e métodos específicos, que deveriam ser definidos e implementados segundo as ferramentas e a semântica da linguagem, ou seja, com o uso dos recursos de construção para a implementação dos conceitos fornecidas pelo Object Pascal, a linguagem de programação do ambiente Kylix e Delphi. Porém, a exemplo do KineGraph, no que se refere a tipo abstrato de dados, conforme observado na seção 3.2, o conceito foi mantido, mas as construções de abstrações da linguagem fonte, as quais permitiriam a implementação de classes e correlatos, não foram utilizadas em sua totalidade.

Com relação ao objetivo do SV, é interessante ressaltar o grande benefício que se obtém, com um custo relativamente baixo, com a disponibilização de diferentes traçados para o usuário, visando facilitar a leitura de uma rede de serviços. Entretanto, deve-se ressaltar também que mesmo trabalhos mais sofisticados em visualização de rede, tais como, HE & EICK (1998) e HE (1999), não se preocupam em desenhar o sistema de outras formas, mas buscam facilitar o entendimento através de outras técnicas, que podem ser aplicadas em conjunto ou isoladamente, tais como, filtros, uso de símbolos e convenções para representação, ou o desenho da rede sobre um contexto visual amigável ao

usuário (seção 2.3). Ambas abordagens querem facilitar o entendimento da rede e não são excludentes, o que permite, em projetos futuros, combinar tais técnicas para possibilitar melhores resultados.

Cabe também observar que apesar de o KitNet ter iniciado com o porte da GrBasic, o KineGraph nunca foi colocado em funcionamento. Apesar de o desenvolvedor saber o quanto seria mais fácil o entendimento de certos procedimentos existentes nesta biblioteca, o fez através da leitura do código fonte da GrBasic, buscando, principalmente, evitar ser influenciado na proposta de interação da ferramenta com o usuário. Entretanto, é notório pela leitura de MARKENZON & VERNET (1997), que muitas das soluções desenvolvidas no KitNet são encontradas no KineGraph com maior eficácia e sofisticação.

5 CONCLUSÕES E TRABALHOS FUTUROS

Em dezembro de 1996, foi publicado o 28º volume da “Computing Surveys”, em comemoração ao cinquentenário da “Association for Computing Machinery” (ACM), com o título “Strategic Directions in Computing Research” (ACM COMPUTING SURVEYS, 1996), objetivando traçar o cenário futuro da pesquisa em computação em diversas áreas da informática, tais como, inteligência artificial, geometria computacional, arquitetura de computadores, banco de dados, comércio eletrônico, métodos formais, interação homem-máquina, programação e linguagens, engenharia de software, redes e telecomunicações, teoria da computação, etc., cujo texto inicial originou-se a partir de um workshop realizado no Massachusetts Institute of Technology, em junho deste mesmo ano, reunindo mais de trezentos pesquisadores, divididos em vinte e dois grupos de trabalho.

Neste trabalho, a visualização foi objeto de estudo dentro das áreas de geometria computacional (TAMASIA et al., 1996), redes e telecomunicações (CLARK et al., 1996), assim como em interação homem-máquina (MYERS et al., 1996). Em alguns casos abordada como ferramenta de apoio de tais áreas e, em outros casos, com o sentido contrário, isto é, como se estas fossem subsídios para a visualização de informações, e, desta forma, foi tratada com maior ou menor especificidade.

As principais contribuições da visualização com as citadas áreas, ou vice-versa, foram observadas tanto em perspectivas futuras quanto em trabalhos já desenvolvidos, destacando-se, por exemplo, na área de redes e telecomunicações, em projeto e validação de redes de serviços e em gerência de redes. Além disso, ressalta-se também a grande inserção, e bastante significativa, em trabalhos de animação de algoritmos, GIS, visibilidade e desenho de grafos, na área de geometria computacional, e, ainda, ensino e disseminação de conhecimento, em elementos de interface e manipulação direta de objetos gráficos, no acesso a informação distribuída, para interação homem-máquina.

Este cenário vem corroborar o exposto ao longo deste texto, mostrando a importância e o crescimento da visualização e sua interação com as demais

áreas, além do que, os atuais problemas de informática apresentam-se com processamento de informações em amplo espectro (SALGADO et al., 1992), não permitindo seu tratamento por uma única disciplina.

O KitNet, o protótipo gerado, por ser protótipo e por não ser o objetivo direto deste trabalho, não é completo, mas não poderia deixar de ser desenvolvido, sob pena de “faltar algo” na compreensão dos problemas reais, e das soluções verdadeiras que podem ser elaboradas no tratamento das informação a serem visualizadas.

O KitNet é prova da interdisciplinaridade exigida na área de informática, pois para o desenvolvimento de um simples protótipo de visualização, foi necessário, em maior ou menor grau, o envolvimento de disciplinas como grafos, algoritmos, estrutura de dados, geometria computacional, visualização científica, redes de comunicação, computação gráfica, técnicas de animação, projeto gráfico e álgebra linear. Por sua vez, a visualização hoje é ferramenta utilizada em engenharia de software e desenvolvimento de frameworks, análise e simulação de redes, tratamento e representação gráfica de grandes volumes de informações, de objetos abstratos e de relações entre objetos, projeto de web para sistemas distribuídos, validação de sistemas (principalmente sistemas baseado em conhecimento), ensino à distância, inteligência artificial e auxílio à aprendizagem de algoritmos, grafos e linguagens de programação.

De uma maneira geral, falta ao protótipo um melhor tratamento dos elementos da interface, como reagrupar os itens dos menus com novo arranjo, e estabelecer maior critério, estabelecer novos parâmetros ou novos procedimentos. Por exemplo, é possível ser mais criterioso e melhorar a precisão de seleção de arestas no procedimento de remoção, ampliar o número de recursos a serem oferecidos com novos procedimento, o que pode não ter limites, e reduzir as restrições impostas nestas versão, como estabelecer novos parâmetros e ampliar o número máximo de níveis que um traçado em níveis pode chegar.

Entretanto, dois pontos podem ser implementados com razoável facilidade e que seriam um grande e novo diferencial para a ferramenta. Primeiro, torná-lo um “front-end” para um analisador de rede, bastando para isso que o analisador se comunicasse com o KitNet através de seus formatos de importação e, o segundo, a inclusão de um sistema inteligente para decidir como uma ligação pode se

comportar e assim seria possível resolver em tempo real a otimização de tráfego, segundo determinados aspectos, conforme sugerido no exemplo visto na seção 4.3, em que foi analisado o comportamento de uma rede através de suas instâncias.

Também como sugestão de trabalhos futuros e complementando a anterior, sugere-se no momento mais como um requinte do que como gênero de necessidade, novas opções de visualização, novas vistas e traçados, que envolvam “multithread”. Seria necessário, por exemplo, para representação ou análise de protocolos de comunicação de forma mais complexa.

Apesar das vantagens ressaltadas com relação a plataforma utilizada, além de ser considerada uma excelente opção, cabe dizer que, ao projetar-se uma ferramenta a partir deste protótipo, deve ser estudada a possibilidade de integração via web, seja por meios dos recursos disponíveis em Kylix/Delphi ou através de uma nova linguagem de desenvolvimento voltada para web, onde sugere-se Perl ou Java.

Ressalta-se que, por uma questão de princípios, toda essa dissertação foi desenvolvida utilizando-se de ferramentas baseadas em ferramentas cuja licença de uso são gratuitas, em sua maioria GPL ou similares, como os já citados GNU/Linux, Kylix Open Source e até mesmo o software gráfico Gimp, versão 1.1.25, para manipulação das imagens e FIG. existentes neste texto.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ACM COMPUTING SURVEYS. ACM 50TH Anniversary Issue: Strategic Directions in Computing Research. ACM, v. 28, n. 4, december 1996.

ALVES, A. D., OLIVEIRA, M. C. F. de, NONATO, L. G.. Interactive Visualization over the WWW. In: BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING – SIBGRAPI 2000, 13., 2000, Gramado. Anais... Los Alamitos: IEEE Computer Society, 2000. p. 259 - 266.

BAECKER, M., SHERMAN, D.. Sorting Out Sorting. In: ACM SIGGRAPH, 1981, Los Altos. Los Altos: Morgan Kaufmann.

BECKER, R. A., EICK, S. G., MILLER, E. O., WILKS, A. R.. Network Visualization. In: INTERNATIONAL SYMPOSIUM ON SPATIAL DATA HANDLING, 4., 1990, Zurich. **Proceedings...** v. 1, 1990, p. 285 - 294.

BECKER, R. A., EICK, S. G., WILKS, A. R.. Visualization Network Data. **IEEE Transactions on Visualization and Computer Graphics**, v. 1, n. 1, p. 16 – 28, march 1995.

BROWN, M.. **Algorithm Animation**. An ACM Distinguished Dissertation 1987. New York: MIT Press, 1988.

BROWN, M. H., SEDGEWICK, R.. A System for Algorithm Animation. **Computer Graphics**, v. 18, n. 3, p. 177 – 186, july 1984.

CÂMARA, G., MEDEIROS, J. S. de. **Revista Fator GIS: GIS para Meio Ambiente**. Instituto Nacional de Pesquisas Espaciais. São José dos Campos: Sagres, 1996. p. 15.

CANTÚ, M.. **Dominando o Delphi 5 – A Bíblia**. São Paulo: Makron Books, 2000.

CARMO, M. B., CUNHA, J. D.. Filtragem e Representação na Visualização de Informação. In: ENCONTRO PORTUGUÊS DE COMPUTAÇÃO GRÁFICA, 8., 1998. **Anais eletrônicos...** Disponível em: <http://automatix.inesc.pt>. Acesso em 15/03/2001.

- CLARK, D. et al.. Strategic Directions in Networks and Telecommunications. **ACM Computing Surveys**, v. 28, n. 4, p. 679 – 690, december, 1996.
- COSTA, L. F., MONTAGNOLI, C.. Máquinas tomam decisões: Reconhecimento de padrões e mineração de dados. **Ciência Hoje**. Rio de Janeiro, v. 30, n. 176, p. 22 - 29, out. 2001.
- COULLARD, C. R., DILWORTH, D. S., OWEN, J. H.. **GIDEN: A Graphical Environment for Network Optimization - Version J-2.0 Beta**. User Guide. November, 1999.
- DOMINGUE, J.. Using Software Visualization Technology in the Validation of Knowledge Based Systems. In: WORKSHOP ON KNOWLEDGE ACQUISITION FOR KNOWLEDGE-BASED SYSTEMS, 9., 1995, Banff, Canadian. **Proceedings...** Banff, Canadian, 1995. Disponível em: <http://kmi.open.ac.uk/people/domingue/pubs/pubs.html>. Acesso em: 15 fev. 2001.
- DOMINGUE, J., MULHOLLAND, P.. Staging Software Visualizations on the Web. In: IEEE SYMPOSIUM ON VISUAL LANGUAGES (VL '97). Capri, Italy, 1997. Disponível em: <http://kmi.open.ac.uk/people/domingue/pubs/pubs.html>. Acesso em 15 mar. 2001.
- DOMINGUE, J., MULHOLLAND, P.. The Internet Software Visualization Laboratory. In: ANNUAL WORKSHOP OF PSYCHOLOGY OF PROGRAMMING INTEREST GROUP (PPIG97), 9., 1997, Sheffield, England. Disponível em: <http://kmi.open.ac.uk/people/domingue/pubs/pubs.html>. Acesso em 15 mar. 2001.
- DOMINGUE, J., PRICE, B. A., EISENSTADT, M.. A Framework for Describing and Implementing Software Visualization Systems. In: GRAPHICS INTERFACE, 1992, Vancouver, Canada. **Proceedings...** Vancouver, Canada. May, 1992. Disponível em: <http://kmi.open.ac.uk/people/domingue/pubs/pubs.html>. Acesso em 15 mar. 2001.
- EICK, S. G., WILLS, G. J.. Navigating Large Networks with Hierarchies. In: IEEE CONFERENCE ON VISUALIZATION. **Proceedings...** 1993, p. 204 - 210.
- EISENSTADT, M., DOMINGUE, J., RAJA, T., MOTTA, E.. Visual Knowledge Engineering. *IEEE Transaction on Software Engineering*. v. 16, n. 10, p. 1164 – 1177.

- FERREIRA, A. B. de H.. **Novo Dicionário Aurélio - Século XXI**. Rio de Janeiro: Nova Fronteira, 1999.
- HE, T.. Internet-Based Front-End to Network Simulator. In: Data Visualization 1999. **Proceedings of Eurographics 99**. May 1999. p. 247 – 252.
- HE, T., EICK, S. G.. Constructing Interactive Network Visual Interfaces. 1998. Disponível em: <http://www.bell-labs.com/user/taosong>. Acesso em 15 jan. 2001.
- HERMAN, I., MELAÇON, G., MARSHALL, M. S.. Graph Visualization and Navigation in Information Visualization: a Survey. **IEEE Transactions on Visualization and Computer Graphics**, v.6,n. 10, 2000.
- HIMSOLT, M.. Graph^{Ed}: An Iterative Graph Editor. In: 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS 89). **Lecture Notes in Computer Science**. v. 349, february 1989.
- IEZZI, G. et al.. **Fundamentos de Matemática Elementar: Complexos Polinômios Equações**. v.6. 2a. ed. São Paulo: Atual, 1977.
- KAMADA, T., KAWAI, S.. A General Framework for Visualizing Abstract Objects and Relations. **ACM Transactions on Graphics**, v. 10, n. 1, p. 1 – 39, january 1991.
- KRISHNAMOORTHY, M. S., SWAMINATHAN, R.. Program Tools for Algorithm Animation. **Software-Pratice and Experience**, v. 19, n. 6, p. 505 – 513, june 1989.
- LEÃO, M.. **Borland Delphi 5: Curso Básico & Rápido**. Rio de Janeiro: Axcel Books do Brasil, 2000.
- LONGO, M., SMITH Jr., R.. **Delphi 4 Total: Dominando a Ferramenta**. Rio de Janeiro: Brasport, 1999.
- MARKENZON, L., VERNET, O.. **Kinegraph: Manual do Usuário - Versão 3.0**. 1997.
- MARKENZON, L., VERNET, O.. The Design and Implementation of Graph Algorithms. In: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND INFORMATICS, 4., 1998. **Proceedings...** 1998. p. 138 - 141.

- MINCY, J. W., THARP, A. J.; TAI, K.. Visualizing Algorithms and Processes with the Aid of a Computer. In: SIGCSE TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 14., 1983, Orlando. **Proceedings...** New York:ACM, 1983. p. 106 - 111.
- MYERS, B.. Visual Programming, Programming By Example and Program Visualization: A Taxonomy. In: CHI'86 HUMAN FACTORS IN COMPUTING SYSTEMS, 1986. **Proceedings...** New York. 1986. p. 59 – 66.
- MYERS, B. et al.. Strategic Directions in Human-Computer Interaction. **ACM Computing Surveys**, v. 28, n. 4, p. 794 - 809, december, 1996.
- PRICE, B. A., BACKERT, R. M., SMALL, I. S.. A Principled Taxonomy of Software Visualization. **Journal of Languages and Computing**, v. 4, p. 211 – 266, 1993.
- QUARESMA, P., LOPES, J. G. P.. Automatização do Desenho de Grafos e sua Aplicação em Inteligência Artificial. Disponível em: <http://www.di.uevora.pt/~pq/papers/>. Acesso em 11 mai. 2002.
- SALGADO, A. C., FONSECA, D., ALBUQUERQUE, E. S. de, MEIRA, S. R. de L.. Sistemas Hipermissão: Hipertexto e Banco de Dados. Porto Alegre: Instituto de Informática da UFRGS, 1992. 194 p.
- SILVA, J. C. G., ASSIS, F. S. G. de. **Linguagens de Programação: Conceitos e Avaliação**. São Paulo: McGraw-Hill, 1988.
- SORENSEN, V.. Missionária da Animação. **Ciência Hoje**, Rio de Janeiro, v. 29, n. 174, ago. 2001. p. 6 – 9. Entrevista.
- SZWARCFITER, J. L.. **Grafos e Algoritmos Computacionais**. 2 ed. Rio de Janeiro: Campus, 1986.
- TAMASIA, R. et al.. Strategic Directions in Computational Geometry. **ACM Computing Surveys**, v. 28, n. 4, p. 591 – 606, december, 1996.
- TANENBAUM, A. S., WOODHULL, A. S.. **Sistemas Operacionais: Projeto e Implementação**. 2a. ed.. Porto Alegre: Bookman, 2000.
- TAYLOR, D., MISCHER, J., GENTRY, T.. **Kylix - Delphi para Linux: Soluções Práticas e Avançadas**. Rio de Janeiro: Campus, 2001.

VERNET, O.. **Kamada**. Rio de Janeiro, RJ, 1993. 1 disquete; 3 ½ pol. Programa aplicativo.

WEGNER, P., DOYLE, J.. Editorial: Strategic Directions in Computer Research. **ACM Computing Surveys**, v. 28, n. 4, p. 565 – 574, december, 1996.

7 ANEXOS

7.1 ANEXO 1: ALGUNS EXEMPLOS DE VISUALIZAÇÃO

- A) Uma rede com 7 pontos dispostos aleatoriamente, em coroa e pelo Kamada, com 8 ligações entre alta, média e baixa capacidade, sem utilização, sobrecarregadas, alta, média e baixa utilização. Exibição sem grade e moldura.

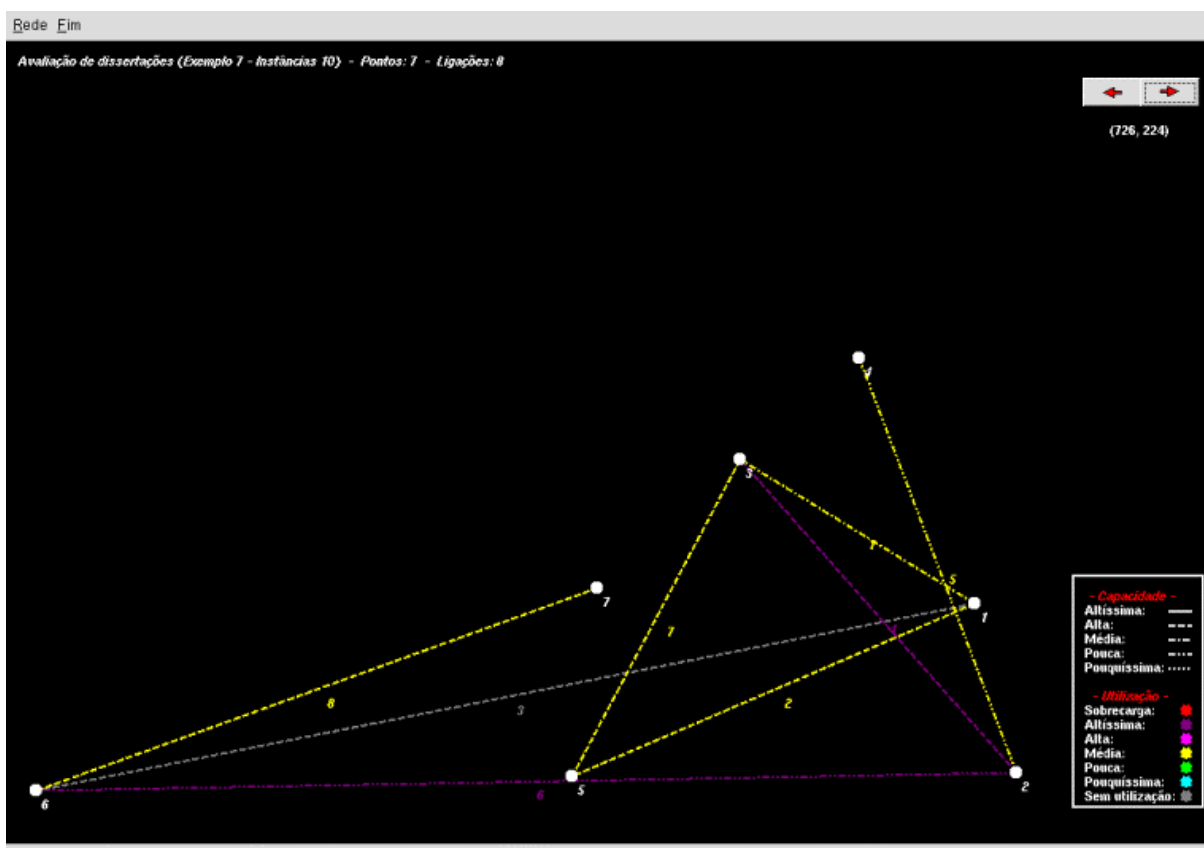


FIG. 7.1.1: Exemplo "A" – Traçado aleatório.

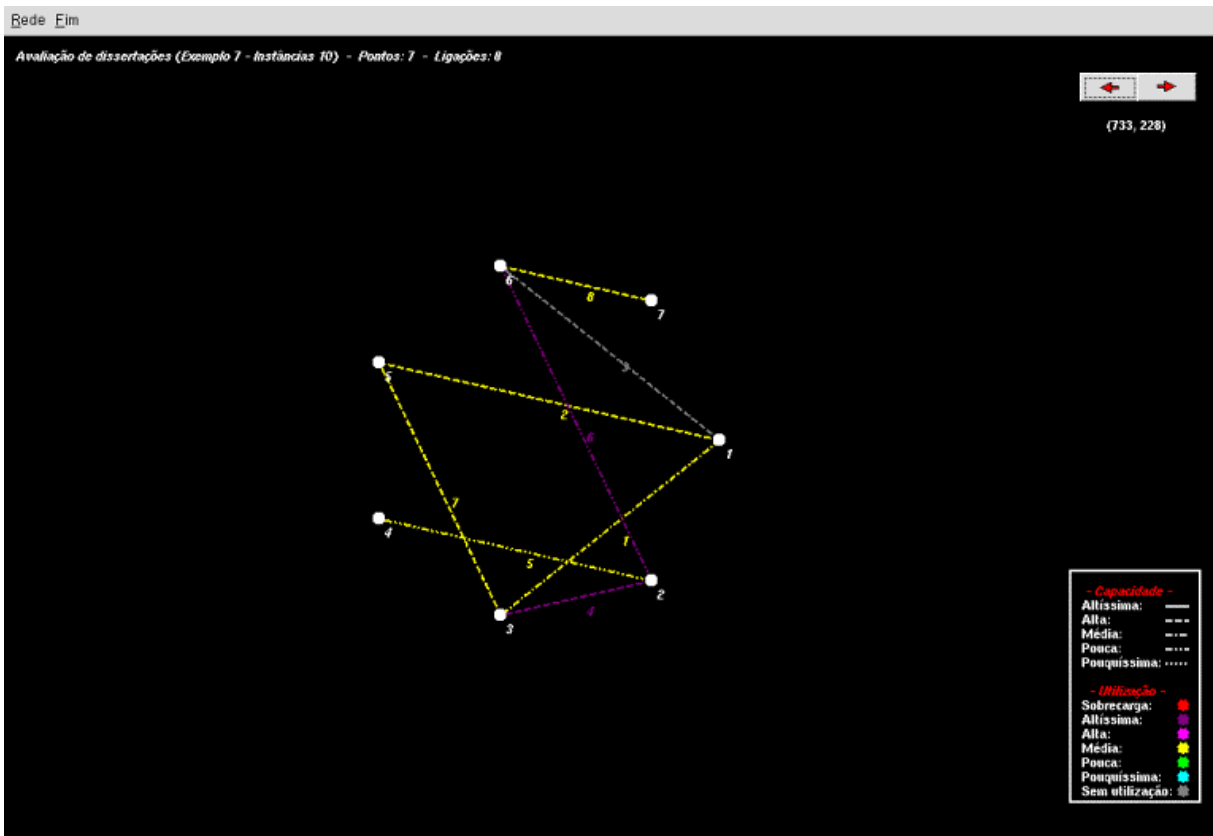


FIG. 7.1.2: Exemplo "A" – Traçado em coroa.

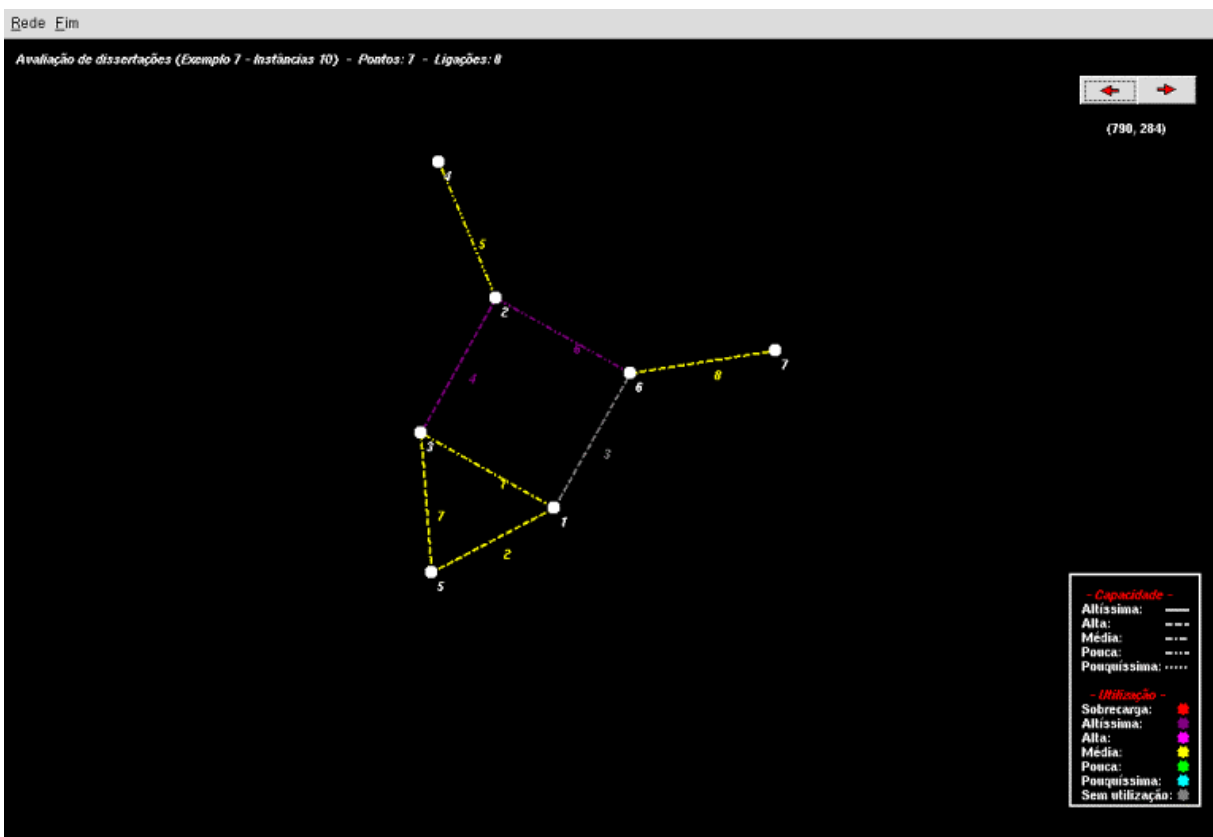


FIG. 7.1.3: Exemplo "A" – Traçado pelo Kamada.

B) Buscas em largura e profundidade em uma rede com 10 pontos dispostos aleatoriamente, em coroa, pelo Kamada e em níveis, com 26 ligações diferenciadas em ligações de caminho e ligações alternativas. Exibição com grade

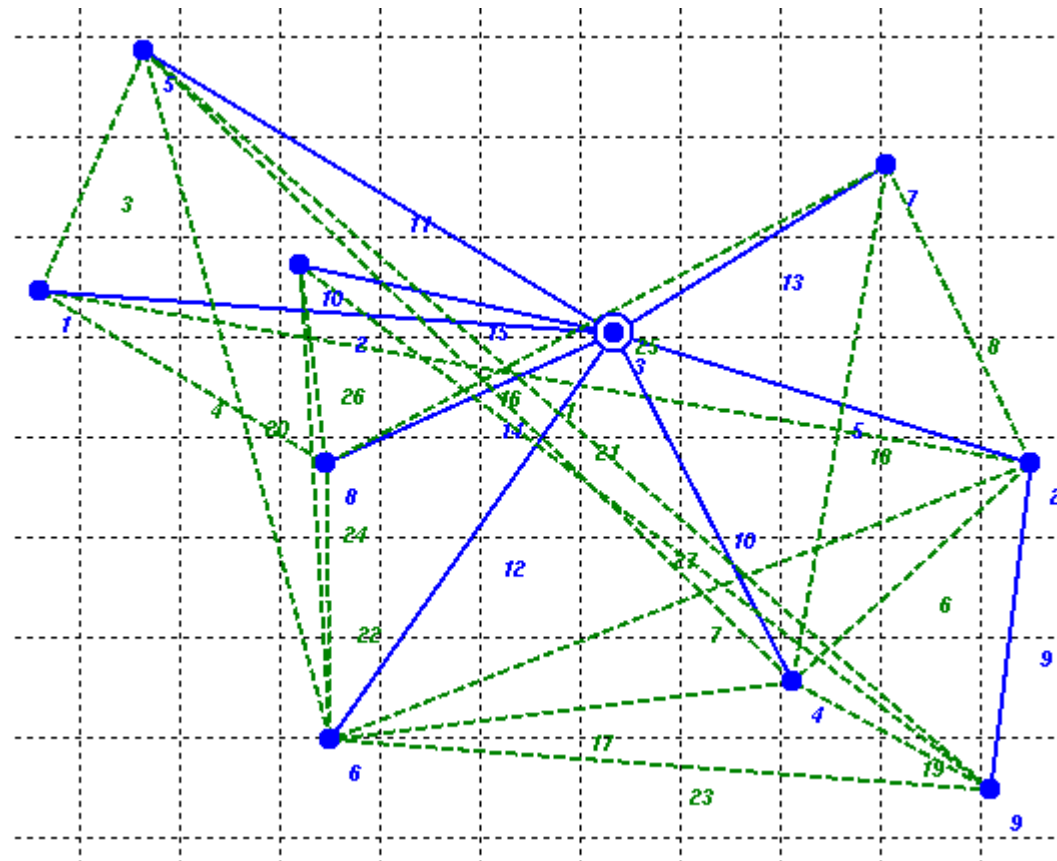


FIG. 7.1.4: Exemplo "B" - Busca em largura sobre traçado aleatório, partindo do ponto 3 e com ligações alternativas visíveis.

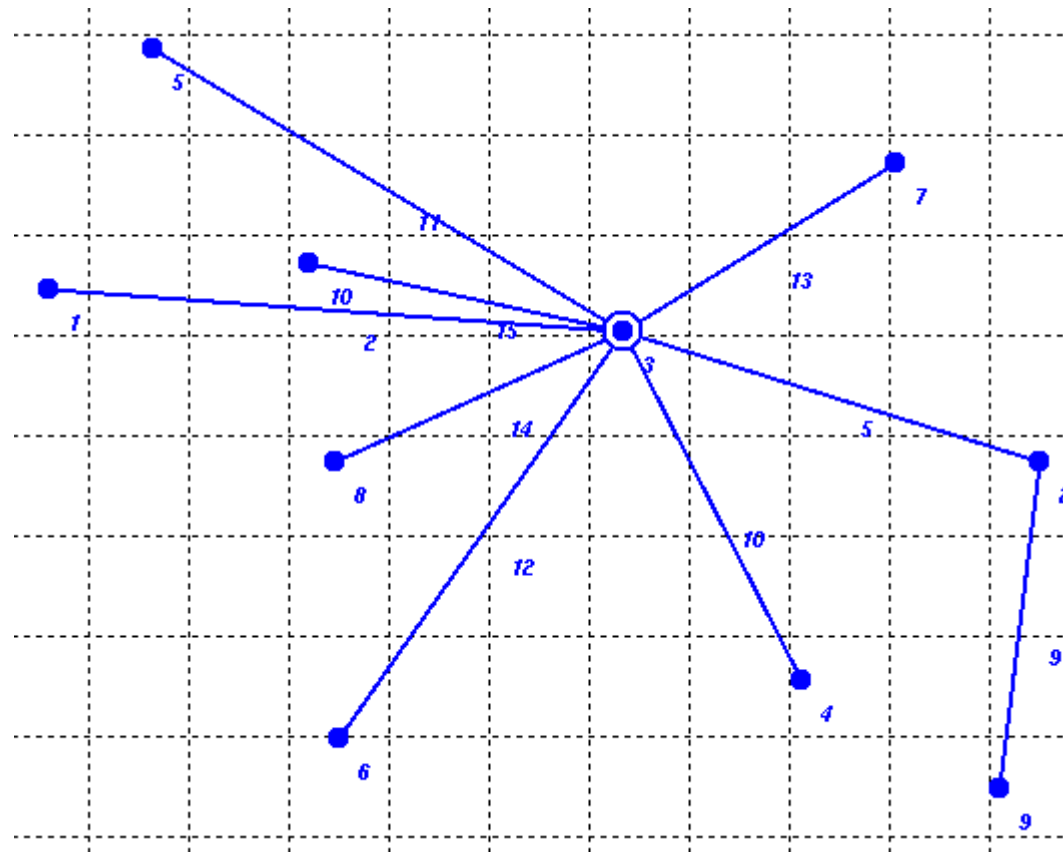


FIG. 7.1.5: Exemplo "B" - Busca em largura sobre traçado aleatório, partindo do ponto 3 e com ligações alternativas não visíveis.

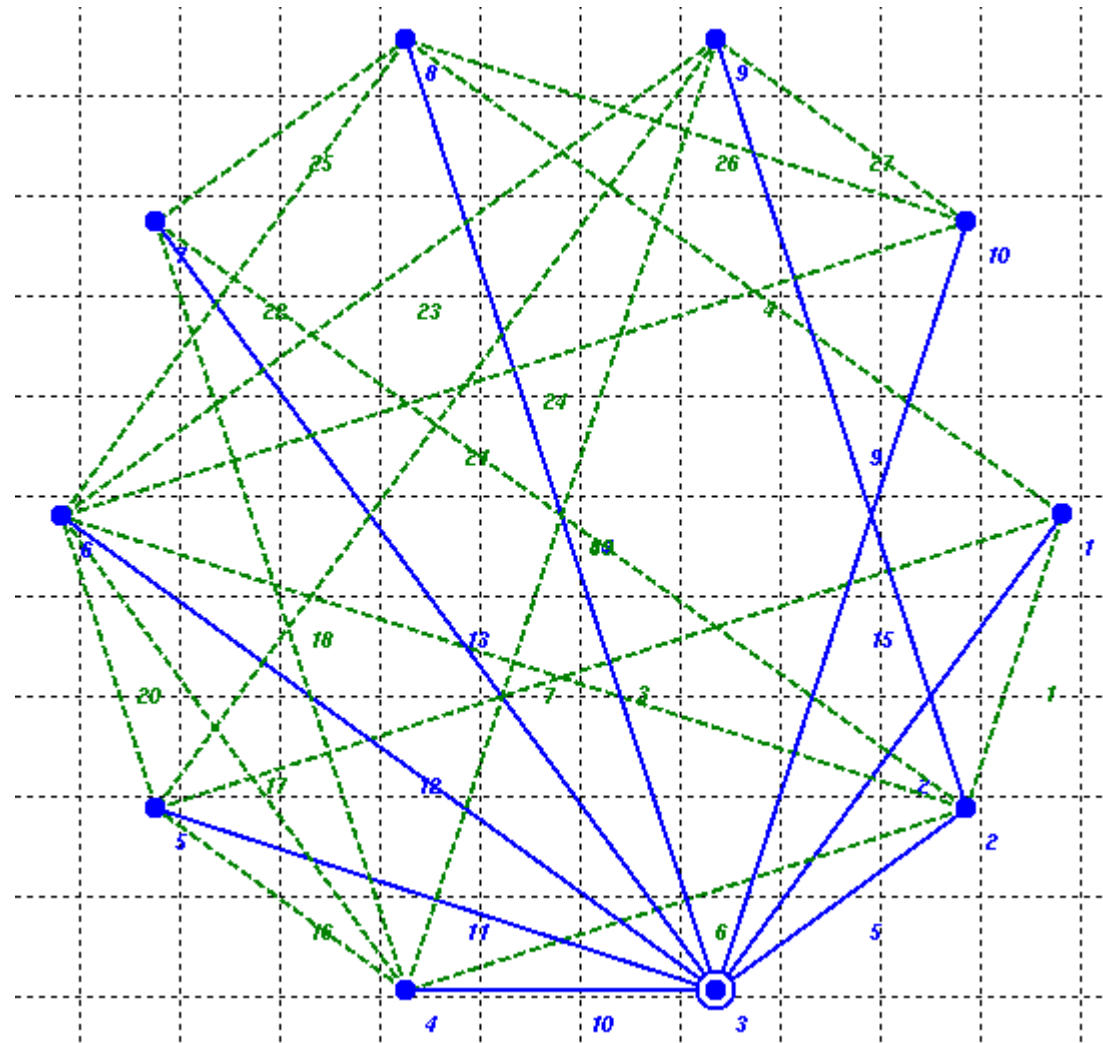


FIG. 7.1.8: Exemplo "B" - Busca em largura sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas visíveis.

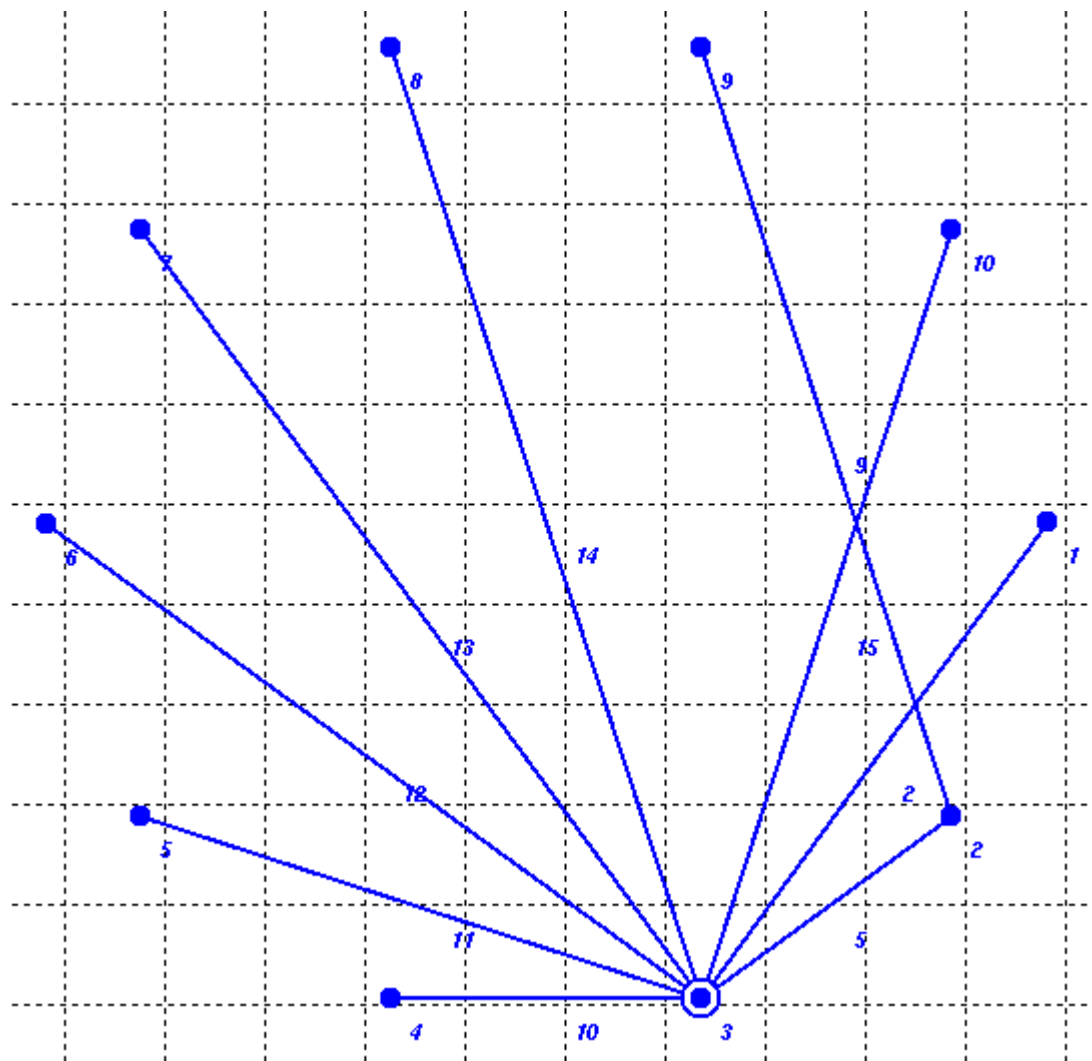


FIG. 7.1.9: Exemplo "B" - Busca em largura sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas não visíveis.

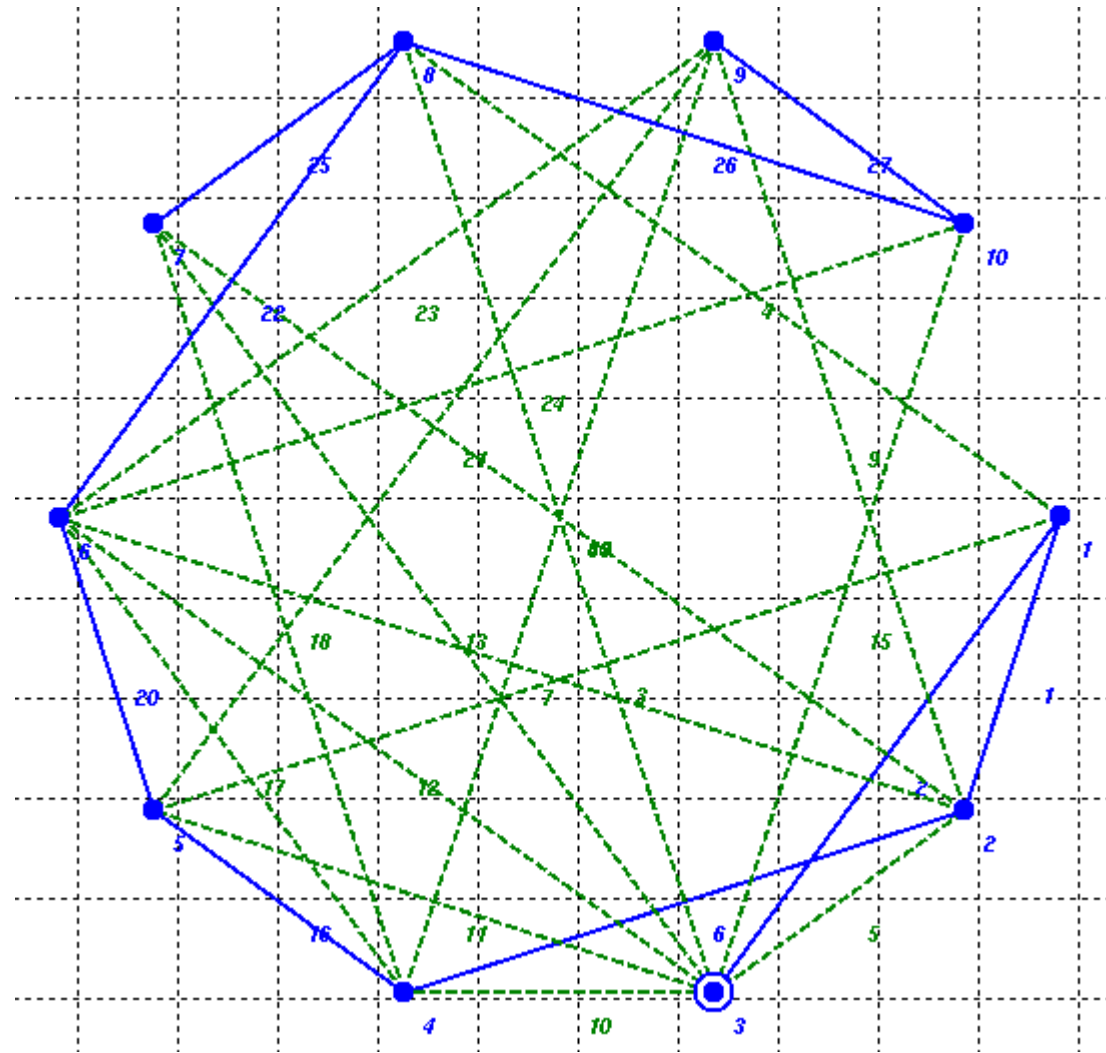


FIG. 7.1.10: Exemplo "B" - Busca em profundidade sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas visíveis.

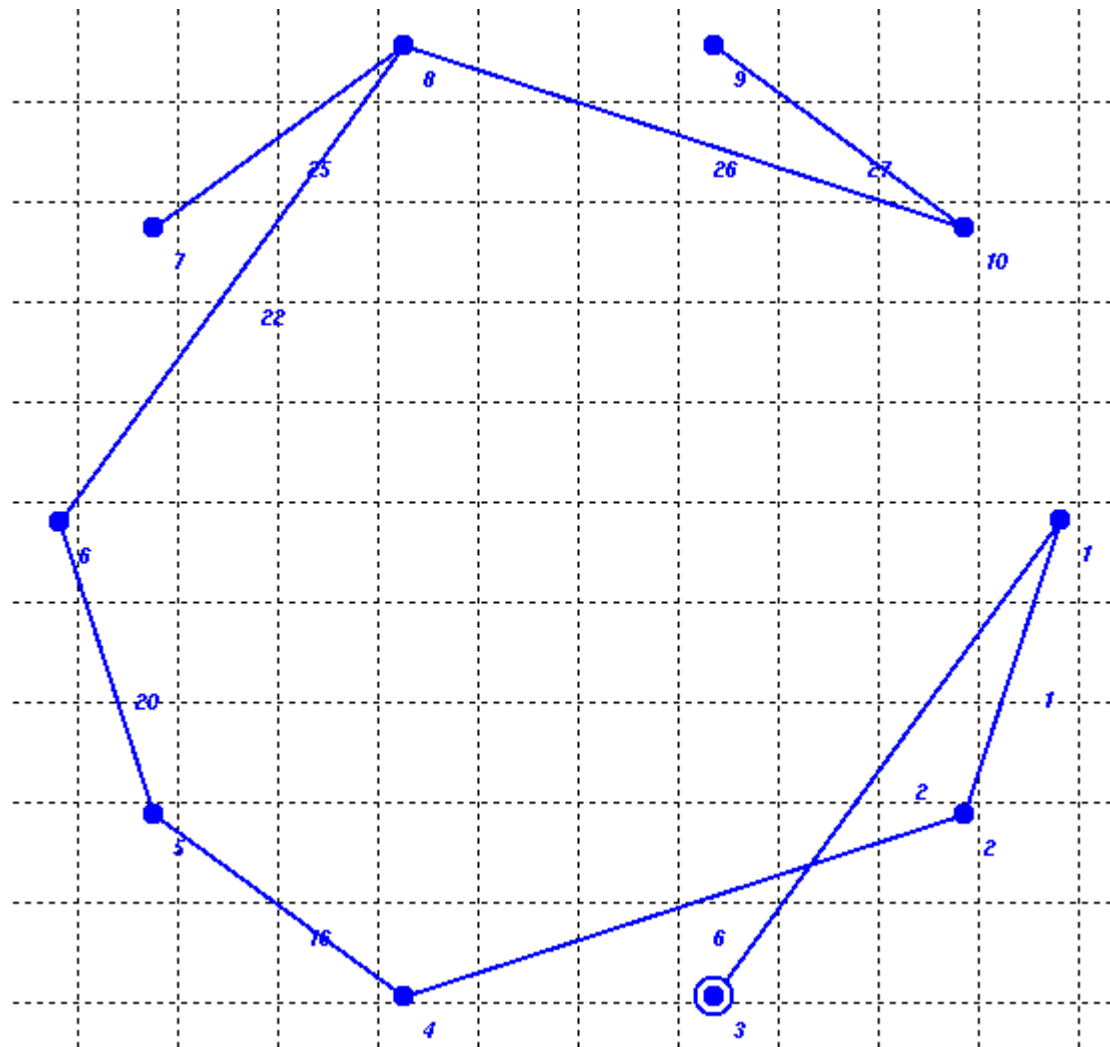


FIG. 7.1.11: Exemplo "B" - Busca em profundidade sobre traçado em coroa, partindo do ponto 3 e com ligações alternativas não visíveis.

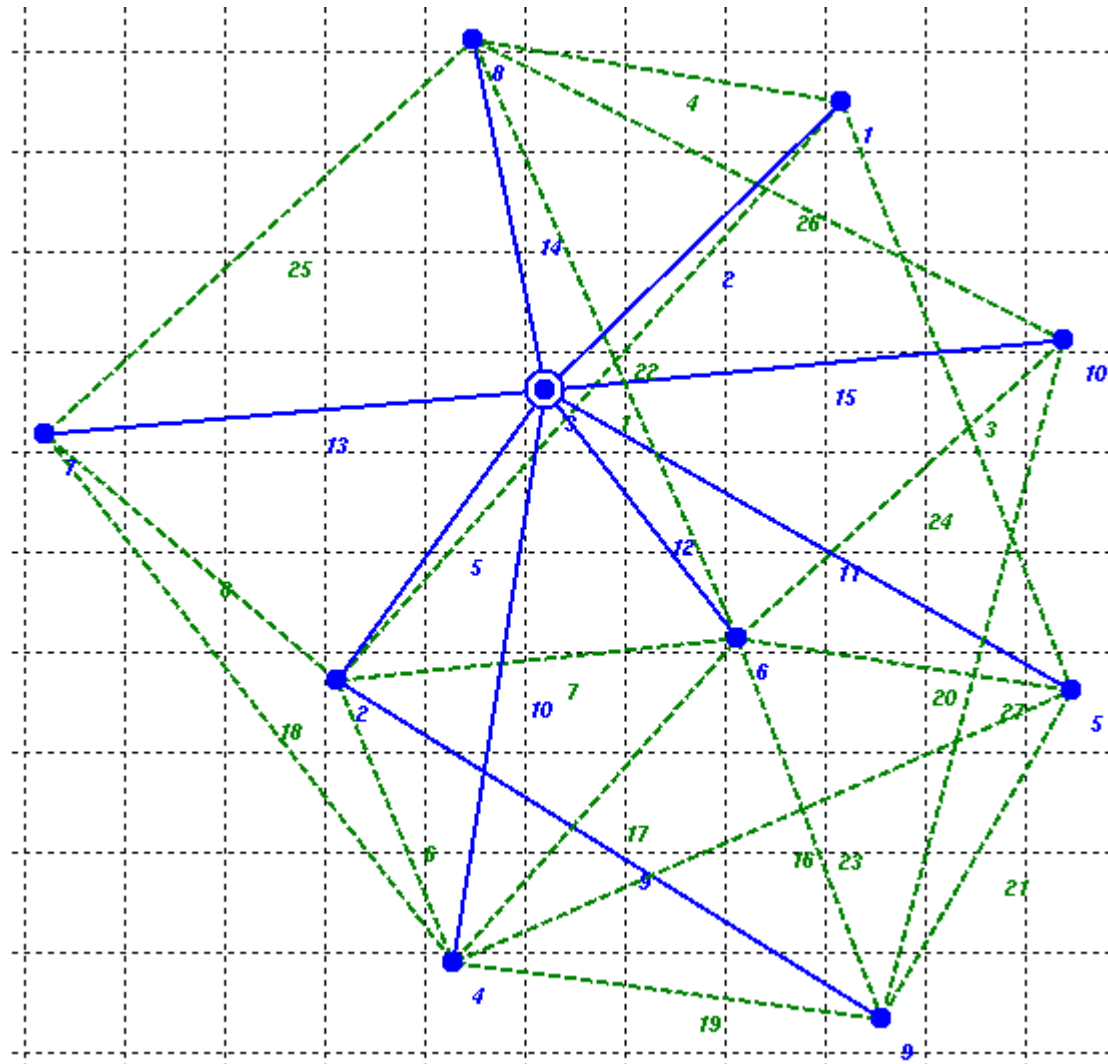


FIG. 7.1.12: Exemplo "B" - Busca em largura sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas visíveis.

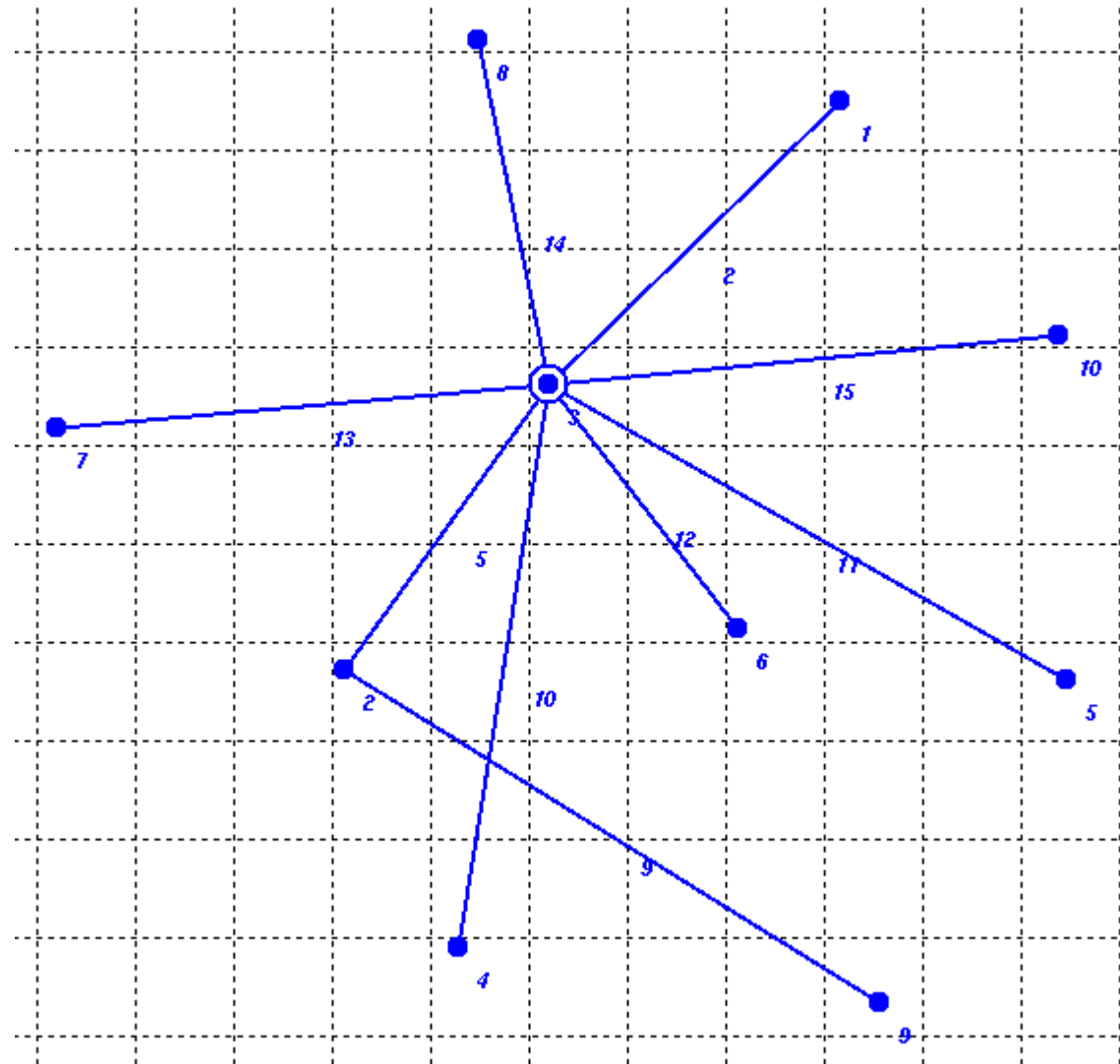


FIG. 7.1.13: Exemplo "B" - Busca em largura sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas não visíveis.

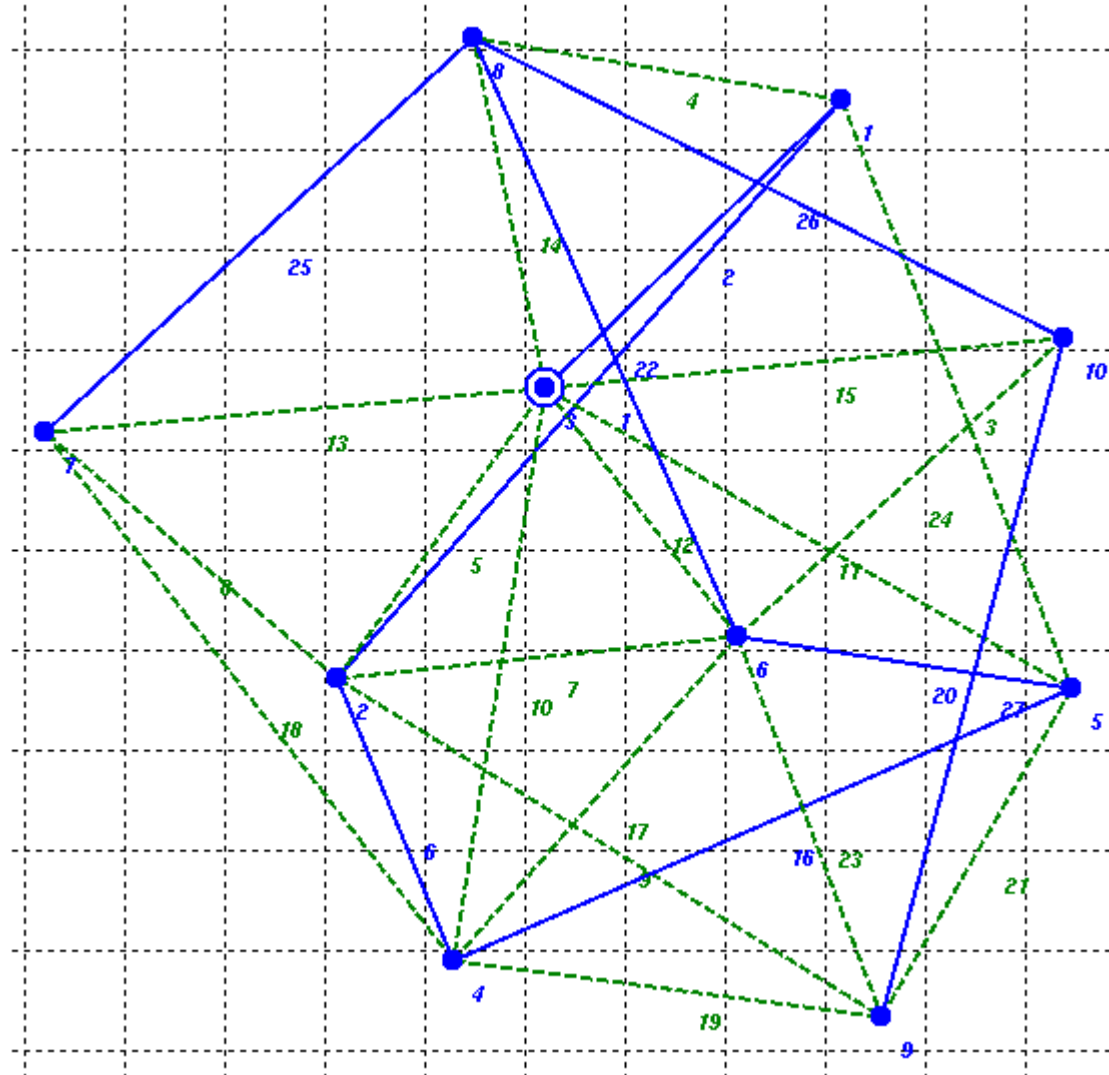


FIG. 7.1.14: Exemplo "B" - Busca em profundidade sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas visíveis.

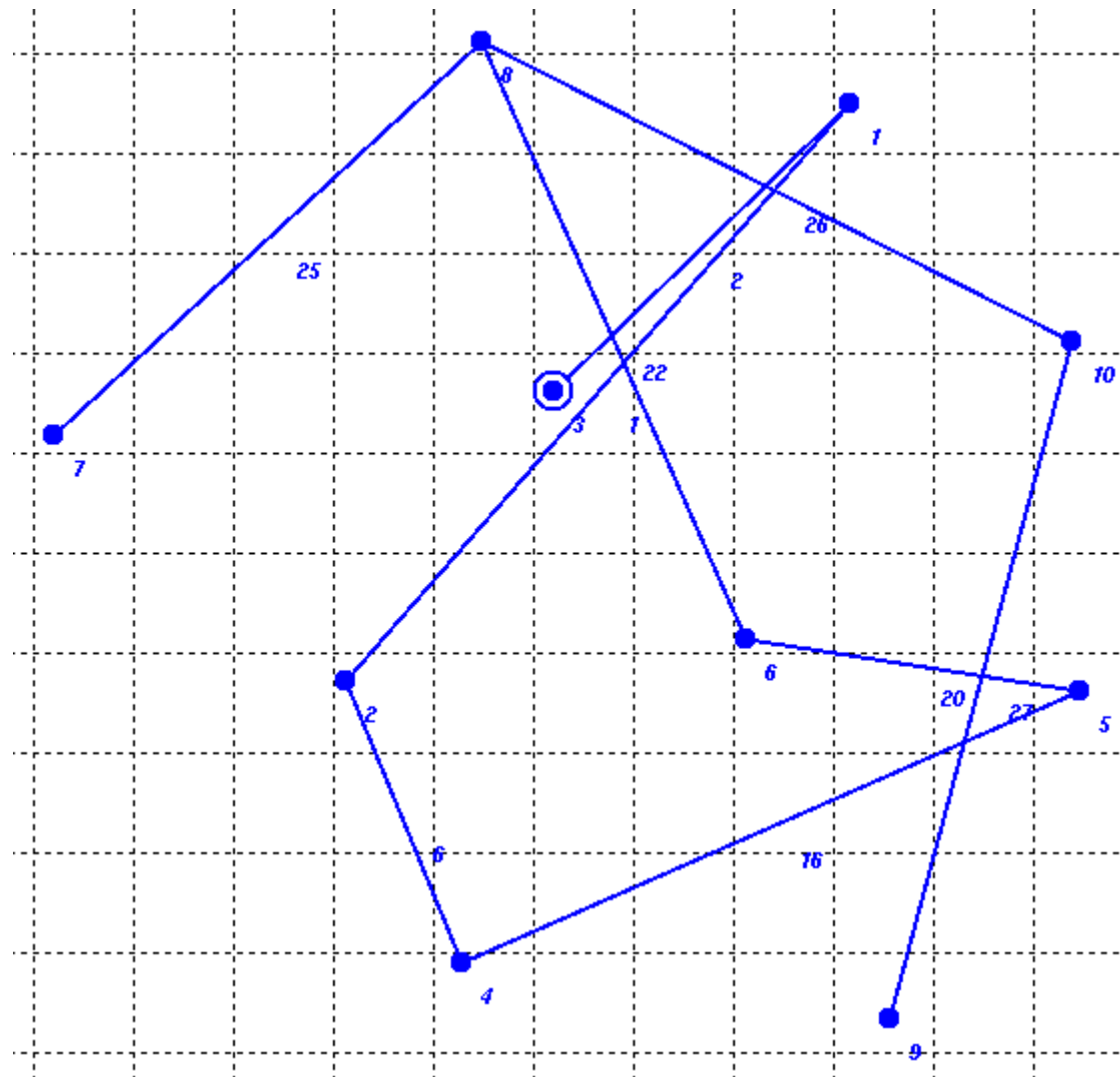


FIG. 7.1.15: Exemplo "B" - Busca em profundidade sobre traçado pelo Kamada, partindo do ponto 3 e com ligações alternativas não visíveis.

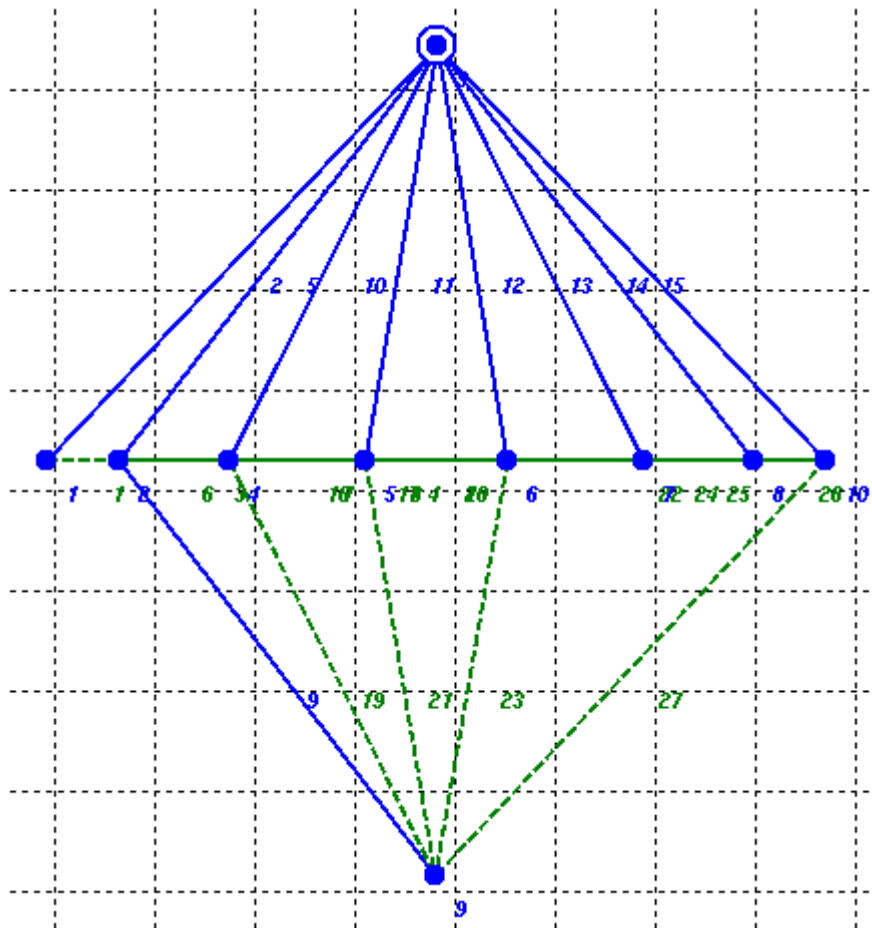


FIG. 7.1.16: Exemplo "B" - Busca em largura sobre traçado em níveis com disposição reta, partindo do ponto 3 e com ligações alternativas visíveis.

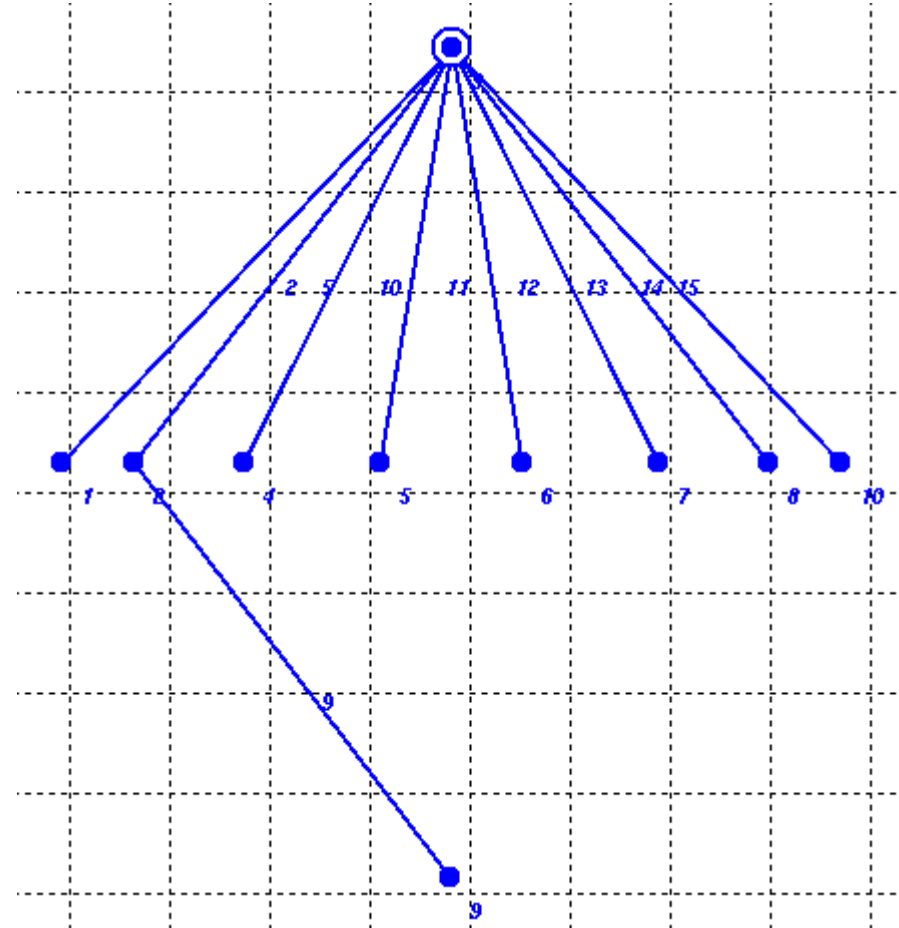


FIG. 7.1.17: Exemplo "B" - Busca em largura sobre traçado em níveis com disposição reta, partindo do ponto 3 e com ligações alternativas não visíveis.

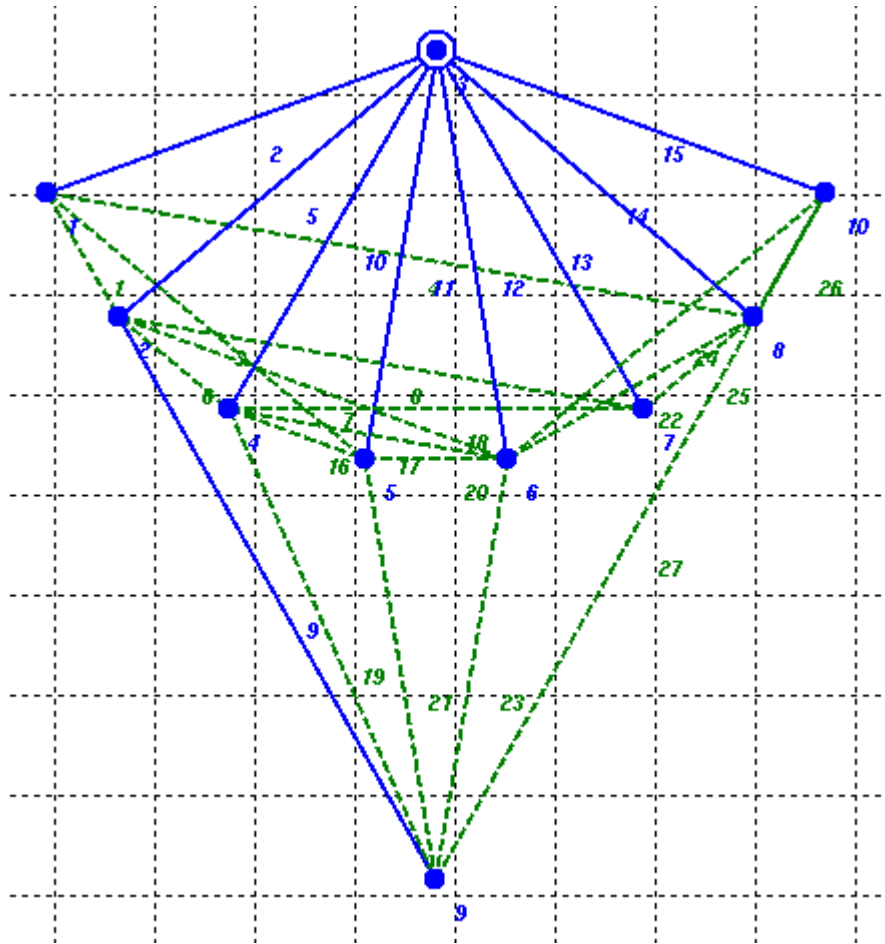


FIG. 7.1.18: Exemplo "B" - Busca em largura sobre traçado em níveis com disposição radial, partindo do ponto 3 e com ligações alternativas visíveis.

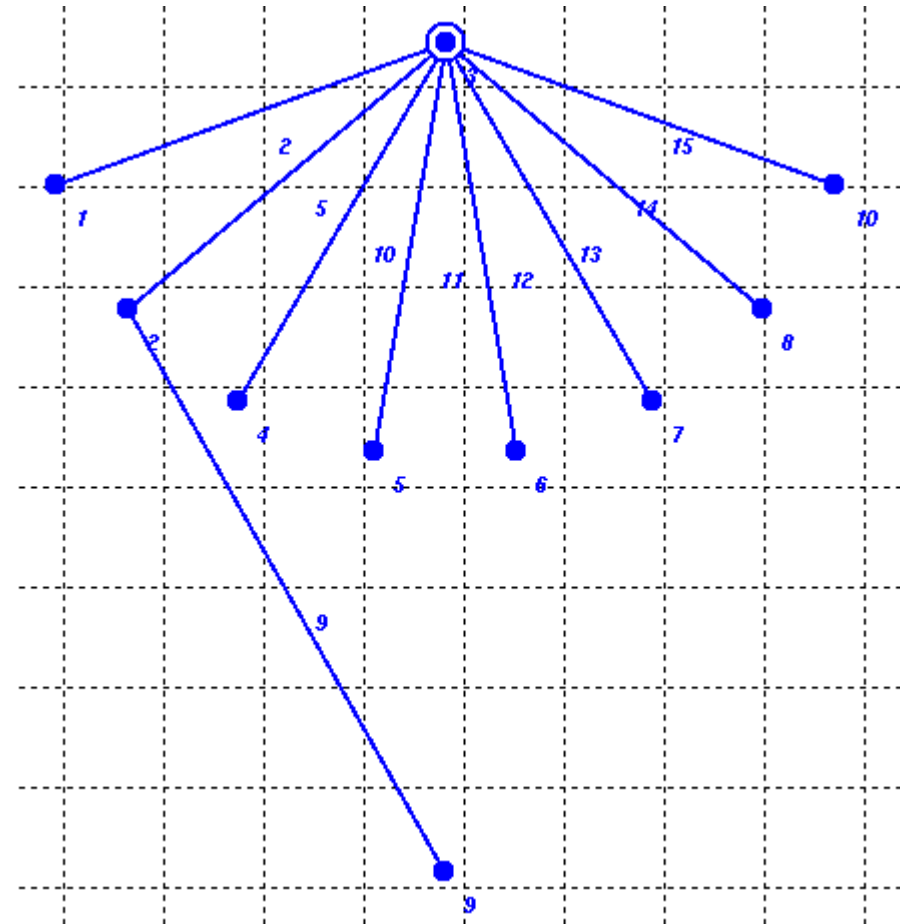


FIG. 7.1.19: Exemplo "B" - Busca em largura sobre traçado em níveis com disposição radial, partindo do ponto 3 e com ligações alternativas não visíveis.

- C) Uma rede obtida através da importação do arquivo reproduzido na seção 4.6, com 35 pontos dispostos aleatoriamente, em coroa, e em níveis, com 49 ligações.

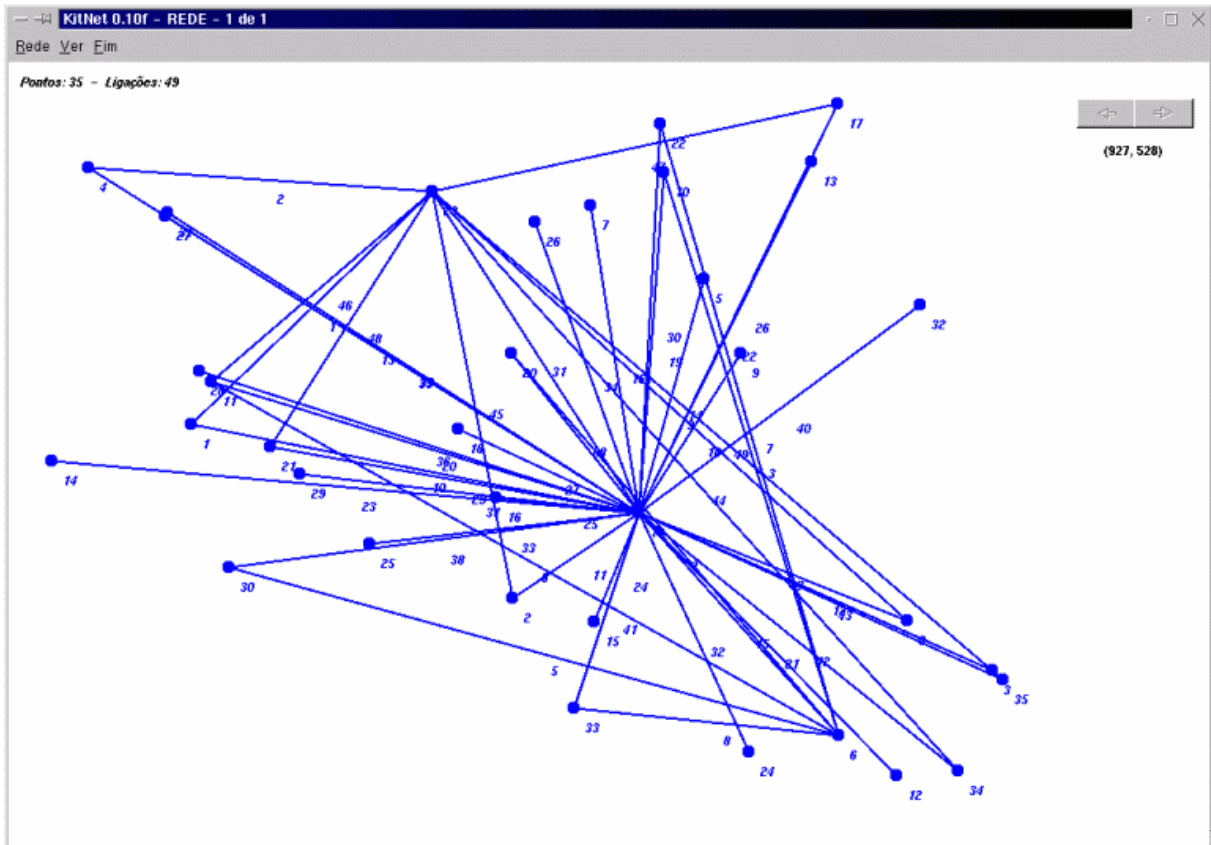


FIG. 7.1.22: Exemplo "C" – Traçado aleatório, exceto pontos cujas coordenadas foram estabelecidas no arquivo original. Exibição sem grade e moldura.

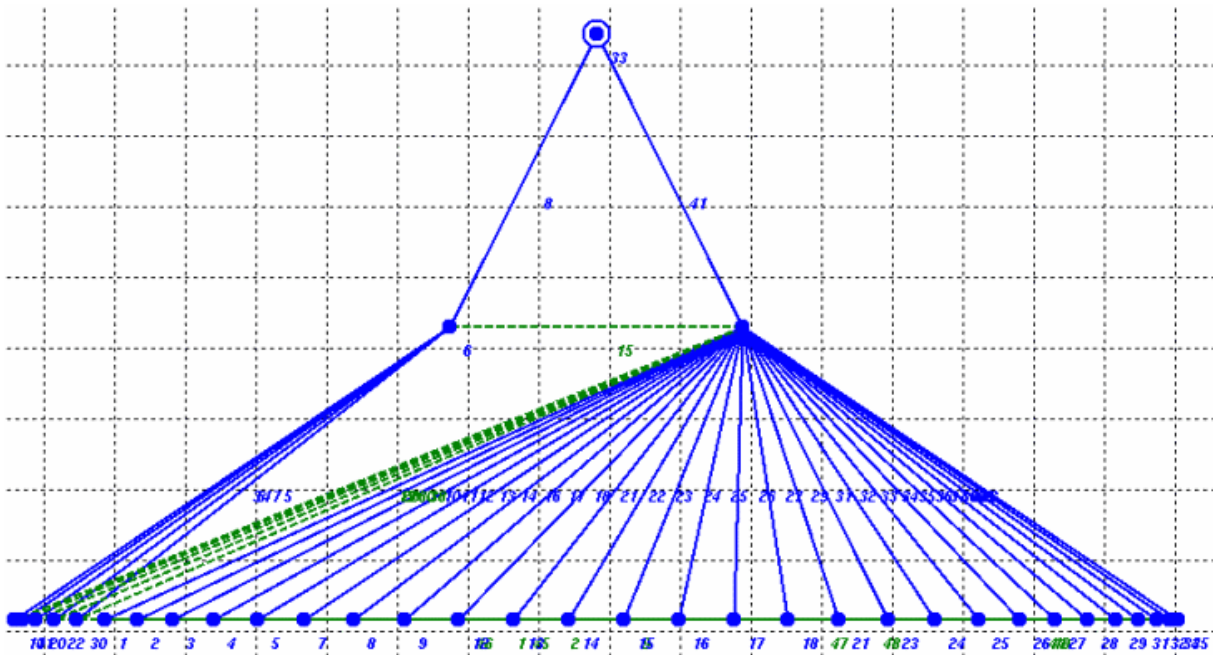


FIG. 7.1.23: Exemplo “C” – Traçado níveis gerado por busca em largura a partir do ponto 33, com ligações alternativas visíveis e posicionamento reto dos pontos. Exibição com grade.

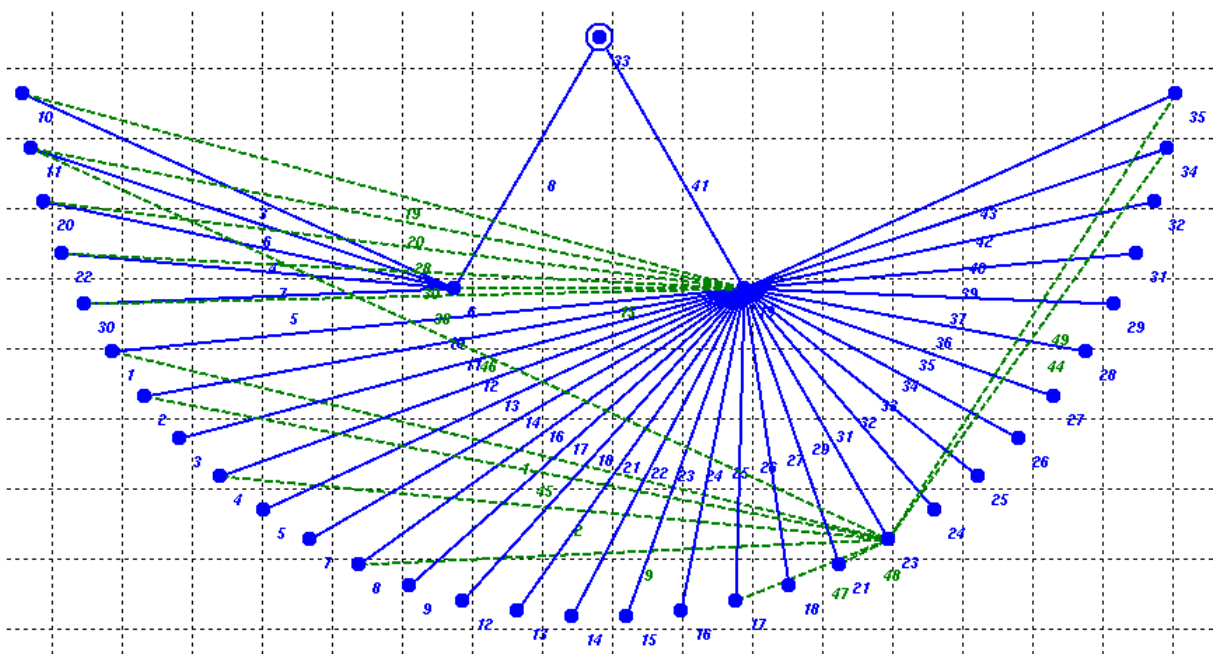


FIG. 7.1.24: Exemplo “C” – Traçado níveis gerado por busca em largura a partir do ponto 33, com ligações alternativas visíveis e posicionamento radial dos pontos. Exibição com grade.

7.2 ANEXO 2: ELEMENTOS E PROCEDIMENTOS ENVOLVIDOS NA VISUALIZAÇÃO DE REDES DE SERVIÇO

Elemento:	1. Rede de Serviço
Descrição/Definição:	<ul style="list-style-type: none"> • $N = (G, I)$
Observações, atributos, características, etc. do elemento:	<ul style="list-style-type: none"> • Toda rede é um grafo, porém a recíproca não é verdadeira. • Uma rede pode conter um multigrafo. • Informações para identificação e caracterização. • Tratamento de subrede a partir de bloco ou subgrafo.
Exemplos:	<ul style="list-style-type: none"> • Rede de petri (HIMSOLT, 1989) • DER (HIMSOLT, 1989) • Diagrama de fluxo (flow chart) (HIMSOLT, 1989) • Grafo de alocação de recursos. • Malha rodoviária, ferroviária, fluvial, marítima ou aérea. • Rede de comunicação de dados, elétrica, telefônica ou para telecomunicações em geral (voz, dados ou imagem). • Mapa cadastral⁹
Observações, atributos, características, etc. dos exemplos:	<ul style="list-style-type: none"> • Nada impede que uma rede possua relações ou fluxos direcionadas e, ao mesmo tempo, outras não direcionadas.

⁹ Um mapa cadastral é aquele em que cada um dos seus elementos é um objeto geográfico, que possui atributos e pode estar associado a várias representações gráficas. Por exemplo, lotes em uma cidade são elementos do espaço geográfico que possuem atributos (localização, valor venal, etc.) e podem ter representações gráficas diferentes em mapas de escalas distintas. (CÂMARA & MEDEIROS, 1996)

Elemento:	2. Grafo
Descrição/Definição:	<ul style="list-style-type: none"> • $G = (V, E)$
Observações, atributos, características, etc. do elemento:	<ul style="list-style-type: none"> • Informações para identificação e caracterização do grafo ou da rede associada. • Generalização dos elementos bloco ou subgrafo, assim, exceto quando explicitado, as informações e procedimentos sobre grafos também aplicam-se a blocos e subgrafos.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Elemento:	3. Vértice
Descrição/Definição:	
Observações, atributos, características, etc. do elemento:	<ul style="list-style-type: none"> • Rótulo. • Físicos e não físicos (HE, 1999) • Informações de identificação (HE, 1999)
Exemplos:	<ul style="list-style-type: none"> • Nó, host ou roteador em uma rede de comunicação de dados. • Site na web (HE, 1999) • Cidade, ou um “ponto”, em uma malha rodoviária, ferroviária, em uma rede pluvial, fluvial, telefônica, de energia, hidráulica, etc. • Um “ponto” em uma rede de serviço. • Um processo ou um recursos em um grafo de alocação de recursos. • Servidor em um SGBD ou em um SGBDD. • IP

Observações, atributos, características, etc. dos exemplos:	<ul style="list-style-type: none"> • Papel em cada grupo a qual pertença (cliente, servidor, cliente e servidor, agente, interceptador). • Se host, então qual o gateway default? (D)¹⁰ • Se roteador, então quais as rotas de saída
---	--

Elemento:	4. Aresta
Descrição/Definição:	
Observações, atributos, características, etc. do elemento:	<ul style="list-style-type: none"> • Rótulo. • Direção. (D) • Métrica: Um ou mais valores, associados, ou não, que atribuem um peso para a aresta. (D) • Características estáticas e dinâmicas (HE, 1999) • Informações de identificação (HE, 1999)
Exemplos:	<ul style="list-style-type: none"> • Link entre dois nós em uma rede de comunicação de dados (CD). • Arco de solicitação ou de atribuição em um grafo de alocação de recursos. • Via pública de tráfego de pedestres ou veículos. • Segmento de estrada ou de ferrovia entre dois pontos. • Tubulação ou cabos, entre dois pontos, de uma rede pluvial, telefônica, de energia elétrica, hidráulica, etc.

¹⁰ E: Valor estático; D: Valor dinâmico.

Observações, atributos, características, etc. dos exemplos:	<ul style="list-style-type: none"> • Largura de banda máxima. (E) • Largura de banda reservada. Pode aceitar "n" reservas até o máximo da largura, ou outra restrição. (D) • Largura de banda disponível para uso. (D) • Largura de banda em uso. (D) • Tráfego entre um cliente e um servidor em um SGBDD. • Direção do tráfego em uma via de Qualquer natureza (ruas, estradas ou ferrovias). (D) • Capacidade do escoamento do tráfego em uma via (E) • Direção do escoamento em uma rede pluvial, elétrica, telefônica, hidráulica, etc. (D)
---	--

Procedimento:	5. Tratamento de bloco; Tratamento de subgrafo. (NI)¹¹
Tipo:	<ul style="list-style-type: none"> • Composto; Circunstancial; Visual; Descontínuo.¹²
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo
Descrição/Definição:	<ul style="list-style-type: none"> • Permite selecionar uma área visualizada, determinando um bloco ou um subgrafo, e reduzir todo o bloco a um único ponto, como se quiséssemos “encolher” a área delimitada. A operação inversa também pode ser feita, ou seja, expandir o ponto restaurando o bloco anteriormente encolhido.
Composição:	<ul style="list-style-type: none"> • Seleção; Crunch; Expansão.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • É um metaprocedimento. • Vértices e arestas que constituem o bloco devem manter suas respectivas informações.

¹¹ NI: Indica que o procedimento não foi implementado no protótipo.

¹² Composto: Indica que o procedimento é uma composição de procedimentos; Simples: Indica que o procedimento sozinho produz o evento desejado; Circunstancial: Indica que o procedimento não altera a estrutura da rede; Estrutural: Indica que o procedimento altera a estrutura da rede; Visual: Indica que o evento reflete na visualização da rede; Descontínuo: Indica que o procedimento é executado uma única vez e para ser executado uma outra vez deve ser novamente selecionado; Contínuo: Indica que o procedimento pode ser executado várias vezes sem a necessidade de selecionar novamente o procedimento.

	<ul style="list-style-type: none"> • Arestas de fronteira devem ser mantidas e as arestas e vértices internos ao bloco não serão mais visualizados. • Um bloco já reduzido pode ser novamente reduzido, caso selecionado em uma nova operação. • Redução do número de variáveis a representar eliminando e/ou agrupando variáveis (CARMO & CUNHA, 1998)
Exemplos:	<ul style="list-style-type: none"> • Subrede em uma rede de CD. • Sistema autônomo em uma rede de CD.
Observações, atributos, características, etc. dos exemplos:	<ul style="list-style-type: none"> • Conjunto de nós/hosts que possuem características semelhantes ou que possuem alguma relação.

Procedimento:	6. Identificação de articulação/Identificação de componente biconexo (NI)
Tipo:	<ul style="list-style-type: none"> • Simples; Circunstancial; Visual; Contínuo.
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • Um vértice em um grafo não direcionado simples e conexo é uma articulação se sua remoção torna o grafo resultante desconexo. • Um grafo é dito biconexo se ele não contém nenhuma articulação.
Composição:	
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • Um componente biconexo é um subgrafo/bloco que possui obrigatoriamente uma articulação. Ou seja, é uma especialização do elemento subgrafo/bloco. • A identificação de articulação pode ser importante em redes de serviço para identificar os pontos frágeis de uma rede. • O algoritmo de busca em profundidade pode ser usada para identificar uma articulação.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	7. Zoom (HE, 1999) (CARMO & CUNHA, 1998) (NI)
Tipo:	<ul style="list-style-type: none"> • Composto; Circunstancial; Visual; Contínuo.
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • Amplia ou reduz visualmente toda a área exposta na moldura de apresentação, promovendo uma mudança de escala que obriga a um reposicionamento da visual do que pode ser visto na moldura de apresentação.
Composição:	<ul style="list-style-type: none"> • Seleção; Mudança de escala; Reposicionamento.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • No caso da ampliação, zoom positivo, a área a ser apresentada na moldura diminui, entretanto, possivelmente, os elementos da rede que ainda estiverem visíveis na moldura serão apresentados com maior riqueza de detalhes e de informações. • No caso de redução, zoom negativo, ocorre o inverso. A área a ser apresentada na moldura será maior, entretanto, possivelmente, os elementos apresentados perderão alguns detalhes e informações. • Altera toda a representação visual da rede e a vista deve ser atualizada para refletir o evento.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	8. Alteração no enquadramento (HE, 1999) (NI)
Tipo:	<ul style="list-style-type: none"> • Simples; Circunstancial; Visual; Contínuo.
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • Alteração do conteúdo exibido na moldura de apresentação. A rede visualizada pode possuir centenas de elementos e, de alguma forma, pode ser impossível visualizá-los todos simultaneamente. Portanto, “desliza-se”, no sentido horizontal ou vertical, a moldura sobre a representação visual da rede, permitindo visualizar a parte da representação corrente dentro da moldura.

Composição:	<ul style="list-style-type: none"> • Reposicionamento
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • Pode ser implementado por barras de rolagem. • Deslocar a moldura que limita a visualização sobre a área visualizada da rede, apresentando, “redesenhando”, a nova parte que se encontra dentro da moldura.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	9. Rotação (HE, 1999) (NI)
Tipo:	<ul style="list-style-type: none"> • Composto; Circunstancial; Visual; Contínuo
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • A partir de um ponto selecionado gira-se a moldura permitindo alterar o posicionamento da parte visualizada dentro da moldura nos sentidos horário e anti-horário.
Composição:	<ul style="list-style-type: none"> • Seleção; Reposicionamento.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • Deslocar a moldura que limita a visualização sobre a área visualizada da rede, apresentando, “redesenhando”, a nova parte que se encontra dentro da moldura.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	10. Seleção (HE, 1999)
Tipo:	<ul style="list-style-type: none"> • Simples; Circunstancial; Visual; Contínuo
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • Dimensiona-se visualmente uma área sobre a representação da rede, selecionando os elementos que se encontram nesta área. • Marcar visualmente um ou mais elementos visíveis na moldura.
Composição:	

Observações, atributos, características, etc. do procedimento:	
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	11. Crunch (HE, 1999) (NI)
Tipo:	<ul style="list-style-type: none"> • Composto; Circunstancial; Visual; Descontínuo;
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • Representação do objeto em maior nível de abstração. • Esconde objetos selecionados em um outro objeto que substitui todos os objetos selecionados. • “Embute” vários objetos em um único objeto.
Composição:	<ul style="list-style-type: none"> • Seleção; Mudança de escala; Reposicionamento.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • A representação visual da rede e a vista deve ser atualizada para refletir o evento.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	12. Expansão (HE, 1999) (NI)
Tipo:	<ul style="list-style-type: none"> • Composto; Circunstancial; Visual; Descontínuo.
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo; Vértice; Aresta.
Descrição/Definição:	<ul style="list-style-type: none"> • Representação do objeto em menor nível de abstração. • Permite a visualização de objetos que, por ventura, podem estar “embutidos” no objeto que sofrerá a ação.
Composição:	<ul style="list-style-type: none"> • Seleção; Mudança de escala; Reposicionamento.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • A vista deve ser atualizada para refletir a ação do procedimento.

etc. do procedimento:	
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	13. Mudança de escala (HE, 1999) (CARMO & CUNHA, 1998) (NI)
Tipo:	<ul style="list-style-type: none"> • Simples; Circunstancial; Visual; Contínuo
Elementos envolvidos:	<ul style="list-style-type: none"> • Grafo.
Descrição/Definição:	<ul style="list-style-type: none"> • Alteração na representação dos objetos pertencentes a rede visualizada, permitindo maior visibilidade em um determinado ponto, podendo reduzir ou aumentar a área contida na moldura de visualização.
Composição:	
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • Pode ocorrer devido a alteração de um atributo dinâmico. Um atributo pode tornar um elemento mais, ou menos, importante, • Pode ocorrer devido alteração no foco de interesse, implicando na alteração da magnitude da sua representação (função de grau de interesse). (CARMO & CUNHA, 1998) • Altera toda a representação visual da rede e a vista deve ser atualizada para refletir o evento.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	14. Reposicionamento (HE, 1999) (CARMO & CUNHA, 1998)
Tipo:	<ul style="list-style-type: none"> • Composto; Circunstancial; Visual; Contínuo
Elementos envolvidos:	<ul style="list-style-type: none"> • Vértices
Descrição/Definição:	<ul style="list-style-type: none"> • Seleciona-se a opção de movimentação, seleciona-se o vértice a ser movido e seleciona-se a nova posição do vértice.

Composição:	<ul style="list-style-type: none"> • Seleção; Inserção.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • A vista deve ser atualizada para refletir a alteração na rede. • Todas as arestas incidentes ao vértice movido sofrerão ação do evento.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	15. Inserção (HE, 1999)
Tipo:	<ul style="list-style-type: none"> • Simples; Estrutural; Visual; Contínuo.
Elementos envolvidos:	<ul style="list-style-type: none"> • Vértices; Arestas.
Descrição/Definição:	<ul style="list-style-type: none"> • Inserir novo elemento na rede.
Composição:	
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • A vista deve ser atualizada para refletir a alteração na rede.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	16. Cópia (HE, 1999) (NI)
Tipo:	<ul style="list-style-type: none"> • Composto; Estrutural; Visual; Contínuo.
Elementos envolvidos:	<ul style="list-style-type: none"> • Vértices; Arestas.
Descrição/Definição:	<ul style="list-style-type: none"> • Seleciona-se a opção de cópia e em seguida os elementos a serem copiados.
Composição:	<ul style="list-style-type: none"> • Seleção; Inserção.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • A vista da rede deve ser atualizada para refletir a alteração na rede. • Todas as arestas incidentes ao vértice copiado sofrerão ação do evento.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	

Procedimento:	17. Deleção (HE, 1999)
Tipo:	<ul style="list-style-type: none"> • Composto; Estrutural; Visual; Contínuo;
Elementos envolvidos:	<ul style="list-style-type: none"> • Vértices; Arestas.
Descrição/Definição:	<ul style="list-style-type: none"> • Seleciona-se a opção de deleção e em seguida os elementos a serem excluídos
Composição:	<ul style="list-style-type: none"> • Seleção.
Observações, atributos, características, etc. do procedimento:	<ul style="list-style-type: none"> • A visão da rede deve ser atualizada para mostrar o efeito do procedimento. • A exclusão de uma aresta envolve somente o seu “desaparecimento”, entretanto a exclusão de um vértice envolve o desaparecimento de todas as arestas incidentes.
Exemplos:	
Observações, atributos, características, etc. dos exemplos:	