UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

ERICSSON SANTANA MARIN

Fluzz

Redes sociais: geração, visualização e buscas que maximizam a probabilidade de influência entre indivíduos

Goiânia
2013

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE DISSERTAÇÃO EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Fluzz – Redes sociais: geração, visualização e buscas que maximizam a probabilidade de influência entre indivíduos

Autor(a): Ericsson Santana Marin

Goiânia, 25 de Fevereiro de 2013.

Ericsson Santana Marin – Autor

Cedric Luiz de Carvalho – Orientador

ERICSSON SANTANA MARIN

Fluzz

Redes sociais: geração, visualização e buscas que maximizam a probabilidade de influência entre indivíduos

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

Área de concentração: Sistemas de Informação.

Orientador: Prof. Cedric Luiz de Carvalho

Goiânia
2013

ERICSSON SANTANA MARIN

Fluzz

Redes sociais: geração, visualização e buscas que maximizam a probabilidade de influência entre indivíduos

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Ciência da Computação, aprovada em 25 de Fevereiro de 2013, pela Banca Examinadora constituída pelos professores:

Prof. Cedric Luiz de Carvalho
Instituto de Informática – UFG
Presidente da Banca

Prof. Thierson Couto Rosa
Instituto de Informática – UFG

Profa. Rita Maria da Silva Julia
Faculdade de Computação – UFU

Ericsson Santana Marin

Graduou-se em Ciência da Computação na PUC Goiás - Pontifícia Universidade Católica de Goiás. Durante e após sua graduação, trabalhou com desenvolvimento de software comercial nas plataformas Microsoft, como Visual Basic 6.0 (VB), Active Server Pages (ASP) e Microsoft .NET. Possui experiência com desenvolvimento Desktop e Web desde 1998. Especializou-se em Qualidade e Gestão de Software também pela PUC Goiás, onde realizou o projeto conceitual de uma rede social *online*. Durante o Mestrado, na Universidade Federal de Goiás (UFG), foi bolsista da CAPES, ministrando seminários e aulas sobre diversos temas vinculados à Inteligência Artificial, como Processamento de Linguagem Natural, Algoritmos Genéticos, Técnicas de Busca, dentre outros. Como trabalho prático de Mestrado, desenvolveu na plataforma Java, uma aplicação multiagente para visualização de redes sociais, capaz de realizar buscas que maximizam a probabilidade de influência entre indivíduos. Atualmente é pesquisador em Ciência das Redes, Inteligência Artificial e entusiasta da Web 3.0.

Dedico este trabalho a toda minha família, pelo apoio e suporte emocional, em especial aos meus pais, que proporcionaram condições favoráveis à realização desta obra.

À minha querida companheira Rose Dália, pelo incentivo inicial de ingresso no mestrado. Sem ela, possivelmente as direções intrínsecas à minha jornada acadêmica, poderiam ter sido diferentes. Rose, *you raise me up*!

Aos matemáticos Leonhard Euler, Paul Erdös e Alfréd Rényi pela inestimável contribuição à sociedade acadêmica, principalmente no referente à Teoria dos Grafos.

Aos sociólogos Anatol Rapoport, também matemático, e à Mark Granovetter por tanto contribuírem para os progressos realizados no entendimento das redes sociais.

Aos físicos e matemáticos Duncan Watts, Steven Strogatz, Mark Newman, Albert Barabási, Réka Albert, Mark Buchanan e Augusto de Franco pelos esforços em transformar a Teoria das Redes em Ciência, e por tentarem de forma incansável desvendar obscuros mistérios da natureza.

Aos cientistas da computação Alan Turing, Oliver Selfridge, John Holland, Marco Dorigo, Luca Maria Gambardella, Warrem McCulloch, Walter Pitts pelos trabalhos criados, inspirados na aprendizagem *bottom-up* da natureza.

Aos cientistas Charles Darwin, Evelyn Fox Keller, Lee Segel, Richard Dawkins e Steven Johnson pelas importantes revelações sobre a evolução das espécies.

Ao meu orientador, Professor Doutor Cedric Luiz de Carvalho, por sua grande colaboração para com os desdobramentos deste projeto. Foi graças à sua pessoa, que ingressei ao mundo da Ciência das Redes, além do universo da Inteligência Artificial. Sem esses dois alicerces fundamentais o trabalho não poderia ter sido construído.

A Deus pela força e inspiração obtida durante toda a elaboração desta obra. Muitos de seus mistérios ainda anseiam por ser elucidados, mas para isso Ele nos propiciou a Ciência, que a cada dia caminha um passo nesta sinuosa direção.

Agradecimentos

A realização deste trabalho só foi possível graças à colaboração e ao apoio de algumas pessoas, às quais transmito os meus sinceros agradecimentos:

Aos meus pais pelo suporte nos inúmeros dias de trabalho laborioso. A presença deles foi meu alicerce onipresente, e por isso, minha dívida será eterna.

Agradeço infinitamente à Rose Dália pelos momentos de compreensão às minhas intempéries emocionais, o que certamente contribuiu com meu êxito. Sou extremamente grato por ter como companheira e incentivadora essa pessoa estimável, de índole tão sublime. Amá-la, é estar mais próximo de Deus. Este trabalho também é seu Rose.

Aos meus companheiros de mestrado, tão fundamentais durante os estudos. Eduardo, Sandino, Max, Francisco, Otávio, Walid, Carlos, Iacer, Gustavo, Mariana, Marcos, Edjalma. Sem a cooperação deles a estrada teria sido bem mais árdua.

Ao amigo Jair Abú Alarcón pelas orientações sobre o *framework JADE*.

Aos cientistas da computação Glen Wasson e Brett Tjaden pela criação do *site The Oracle of Kevin Bacon*, fonte de inspiração para os contornos deste projeto.

Ao instrutor de Java Bruno Zafalão da 3Way Networks, imprescindível para que meu conhecimento sobre a plataforma fosse sedimentado.

Aos pesquisadores Joshua O'Madadhain, Danyel Fisher e Tom Nelson pela criação e disponibilização do *framework JUNG*, amplamente utilizado neste trabalho.

A Augusto de Franco por idealizar e viabilizar a Escola de Redes, comunidade virtual fundamental para minhas investigações, pesquisas e questionamentos sobre redes.

À Raquel Recuero, que por sua obra Redes Sociais na Internet, serviu como minha fonte de pesquisa inicial, abrindo portas por onde transitei durante a pesquisa.

À CAPES, pelo importante incentivo ao me conceder a bolsa de estudos.

Aos meus professores de mestrado, Thierson, João Carlos, Diane, Kleber, Les e Telma, que tiveram fundamental importância na minha formação acadêmica.

Ao meu orientador, Professor Doutor Cedric Luiz de Carvalho, pelo suporte constante oferecido. Agradeço pela paciência quando não fui objetivo, pela participação quando não fui eficaz, e especialmente, pela cordialidade quando me faltou inspiração.

A Deus pelo dom da vida e pela dádiva do pensamento.

... só porque os mistérios da era da conectividade frequentemente parecem incompreensíveis, não significa que realmente o sejam.

Duncan Watts,
Físico, sociólogo, pesquisador da Microsoft, e um dos principais arquitetos da Ciência das Redes.

Resumo

Marin, Ericsson S.. **Fluzz**. Goiânia, 2013. 183p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

O propósito desta dissertação é a realização de um estudo sobre redes, mais especificamente das redes sociais, visando à criação de modelos, técnicas e ferramentas para simular a sua geração, produzir o seu mapeamento estrutural gráfico, e viabilizar a realização de buscas nos ambientes que as representam. A geração e visualização de redes, ou grafos, assim como a realização de buscas, compõem um ramo continuamente estudado por diversos profissionais que ingressaram recentemente em uma nova ciência interdisciplinar, enraizada em pesquisas sociológicas e na Teoria dos Grafos: a Ciência das Redes. Alguns resultados provenientes das pesquisas desta ciência subverteram conceitos previamente definidos, e apresentaram características reveladoras sobre o universo social interconectado. Dentre eles destacam-se a desmistificação dos seis graus de separação com a comprovação do fenômeno do "mundo pequeno". Tais revelações inserem-se particularmente no estudo das redes sociais, mas demonstraram ser uma característica onipresente nas demais redes pesquisadas pelos cientistas. Consequentemente, as pesquisas sobre redes sociais principiaram o estudo sobre redes de forma mais geral, produzindo inúmeros trabalhos sobre o tema. Dentro deste contexto, a aplicação *Fluzz*, engendrada sob os alicerces dos Sistemas Multiagentes e dos princípios da contemporânea Ciência das Redes, é introduzida para prover ferramentas de visualização e de simulação da geração de redes sociais, baseadas nos modelos já propostos na literatura e em um novo modelo concebido nesta obra. Para o processo de busca, agentes de *software* capazes de atuar de forma distribuída e paralela, foram implementados utilizando-se de diversas abordagens intrínsecas à Inteligência Artificial, como Algoritmo de Dijkstra, Otimização por Colônia de Formigas e Algoritmos Genéticos. Tais agentes foram projetados para maximizar a probabilidade de influência entre os indivíduos das redes sociais, contribuindo para o aumento da atividade de conexão, de interação, e consequentemente de cooperação entre os mesmos.

Palavras–chave

Ciência das Redes, Redes Sociais, Otimização, Visualização da Informação, Sistemas Multiagentes, Algoritmos de Busca de Caminhos

Abstract

Marin, Ericsson S.. **Social networks: generation, visualization and search that maximize the probability of influence between individuals**. Goiânia, 2013. 183p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

The purpose of this dissertation is to conduct a study of networks, specifically social networks, aimed at creating models, tools and techniques to simulate their generation, to produce their structural graphical mapping, and to enable conducting searches in the environments that represent them. The generation and visualization of networks or graphs, along with conducting searches, compose a branch continuously studied by several professionals who have joined recently in a new interdisciplinary science, rooted in sociological research and Graph Theory: the Science of Networks. Some results from the research of this science subverted concepts previously defined and presented revealing characteristics about the interconnected social universe. Among them stand the demystification of six degrees of separation with proof of the "small world" phenomenon. Such revelations are inserted particularly in the study of social networks, however they have proved to be an ubiquitous feature on other networks surveyed by the scientists. Consequently, research on social networks made the study on networks more general, producing numerous works regarding the subject. Within this context, the application *Fluzz* engendered under the foundations of Multiagent Systems and principles of contemporary Science of Networks, is introduced to provide tools to visualize and simulate the generation of social networks, based on models already proposed in literature and a new model designed in this work. For the search process, software agents capable of acting in distributed and parallel ways were implemented using several approaches intrinsic to Artificial Intelligence such as Dijkstra's Algorithm, Ant Colony Optimization, and Genetic Algorithms. These agents have been designed to maximize the probability of influence of the social network's members, contributing to the increased activity of connection, interaction, and therefore cooperation between them.

Keywords

Science of Networks, Social Networks, Optimization, Information Visualization, Multiagent Systems, Search Paths' Algorithms

Sumário

Lista de Figuras **13**

Lista de Tabelas **16**

Lista de Algoritmos **17**

Lista de Códigos de Programas **18**

1 Introdução **19**
- 1.1 Motivação 21
- 1.2 Objetivos 23
- 1.3 Metodologia 25
- 1.4 Organização da Dissertação 26

2 Ciência das Redes **27**
- 2.1 Complexidade e Emergência 27
- 2.2 Grafos ou Redes 29
- 2.3 Redes Sociais 30
  - 2.3.1 Redes Sociais *Online* 32
- 2.4 Seis Graus de Separação 34
- 2.5 Três Graus de Influência 36
- 2.6 Modelagem de Redes 37
  - 2.6.1 Universo Randômico 37
  - 2.6.2 Não Aleatoriedade 40
  - 2.6.3 Pontes entre Mundos 41
  - 2.6.4 O Modelo Alfa 44
  - 2.6.5 Redes de Mundo Pequeno 46
  - 2.6.6 Redes Sem Escala 50
  - 2.6.7 Redes por Afiliação 55

3 Algoritmos de Busca de Caminhos **59**
- 3.1 Problema do Caminho Mínimo 60
- 3.2 Algoritmo de Dijkstra 62
- 3.3 Otimização por Colônia de Formigas 66
- 3.4 Algoritmos Genéticos 71

4 Agentes e Sistemas Multiagentes **78**
- 4.1 Agentes 79
- 4.2 Sistemas Multiagentes 82

5 Visualização da Informação **84**
5.1 Técnicas de Visualização de Informações 90

6 *Frameworks* Multiagentes Java **91**
6.1 JADE - *Java Agent DEvelopment framework* 92
6.1.1 Arquitetura do JADE 93
6.1.2 Ciclo de Vida de Agentes 94
6.1.3 Tarefas de Agentes 95
6.1.4 Comunicação entre Agentes 96

7 *Frameworks* Gráficos Java **97**
7.1 JUNG - *Java Universal Network/Graph Framework* 98
7.1.1 Arquitetura do JUNG 99
7.1.2 Criação de Grafos 100
7.1.3 Visualização de Grafos 101
7.1.4 Filtros 102
7.1.5 Algoritmos 102
7.1.6 Recursos de Interação 103

8 Análise de Trabalhos Correlatos **105**
8.1 The Oracle of Kevin Bacon 105
8.2 Vizster 107
8.3 SocNetV 108
8.4 Facebook Visualiser, TagGraph, Mentionmapp 109

9 Projeto da Aplicação **111**
9.1 Requisitos Funcionais 112
9.1.1 Persistência dos dados gerados em um banco de dados 112
9.1.2 Visualização bidimensional das redes por meio de grafos 112
9.1.3 Distinção visual dos vértices pela intensidade de cores 112
9.1.4 Seleção individual dos vértices e das arestas 112
9.1.5 Distinção de cores para vértices e arestas selecionados 113
9.1.6 Movimentação individual ou em grupo dos vértices 113
9.1.7 Movimentação do grafo inteiro 113
9.1.8 Variação da escala de visualização do grafo 113
9.1.9 Rotação do grafo em torno de seus eixos 113
9.1.10 Deformação do grafo de maneira uniforme 113
9.1.11 Visualização do grafo por meio de diversos *layouts* 113
9.1.12 Cálculo automático dos pesos das arestas 114
9.1.13 Visualização dos pesos das arestas de forma opcional 114
9.1.14 Visualização do perfil de uma pessoa 114
9.1.15 Visualização do perfil de uma relação de amizade 115
9.1.16 Apresentação da quantidade de vértices do grafo 115
9.1.17 Apresentação da quantidade de arestas do grafo 115
9.1.18 Apresentação do coeficiente de aglomeração da rede 115
9.1.19 Apresentação da distância geodésica média da rede 115
9.1.20 Geração de redes através dos modelos da literatura 115
9.1.21 Geração de redes de acordo com um novo modelo 116

9.1.22 Importação dos dados de redes pré-definidas 118
9.1.23 Adição de conexões às redes existentes 118
9.1.24 Detecção de caminhos entre duas pessoas 118
9.1.25 Detecção de caminhos entre uma pessoa e várias outras 120
9.1.26 Criação de filtros para visualização dos caminhos 120
9.1.27 Salvamento da imagem da rede gerada 120
9.1.28 Deleção da base de dados 121
9.2 Requisitos Não Funcionais 121
9.3 Diagramas de Caso de Uso 121
9.4 Modelagem da Estrutura de Dados da Aplicação 123
9.5 Arquitetura da Aplicação 124
9.5.1 O agentePrincipal 124
9.5.2 A Sociedade de Agentes de Pesquisa 125
9.5.3 O Repositório de Dados 129

10 Desenvolvimento da Aplicação **130**
10.1 Tecnologias Empregadas 130
10.1.1 Java 7 130
10.1.2 Eclipse Indigo 131
10.1.3 JADE 3.7 131
10.1.4 JUNG 4.2 132
10.1.5 PostgreSQL 9.0.1 132
10.1.6 Sistema Operacional Windows 7 133
10.1.7 Arquitetura MVC (Modelo-Visão-Controle) 134
10.1.8 Configuração de *Hardware* 134
10.2 Implementação dos Requisitos 135
10.2.1 RF: Persistência dos dados gerados em um banco de dados 136
10.2.2 RF: Visualização bidimensional das redes por meio de grafos 137
10.2.3 RF: Distinção visual dos vértices pela intensidade de cores 137
10.2.4 RF: Seleção individual dos vértices e das arestas 138
10.2.5 RF: Distinção de cores para vértices e arestas selecionados 139
10.2.6 RF: Movimentação individual ou em grupo dos vértices 139
10.2.7 RF: Movimentação do grafo inteiro 139
10.2.8 RF: Variação da escala de visualização do grafo 139
10.2.9 RF: Rotação do grafo em torno de seus eixos 140
10.2.10 RF: Deformação do grafo de maneira uniforme 140
10.2.11 RF: Visualização do grafo por meio de diversos *layouts* 141
10.2.12 RF: Cálculo automático dos pesos das arestas 142
10.2.13 RF: Visualização dos pesos das arestas de forma opcional 143
10.2.14 RF: Visualização do perfil de uma pessoa 144
10.2.15 RF: Visualização do perfil de uma relação de amizade 145
10.2.16 RF: Apresentação da quantidade de vértices do grafo 145
10.2.17 RF: Apresentação da quantidade de arestas do grafo 145
10.2.18 RF: Apresentação do coeficiente de aglomeração da rede 145
10.2.19 RF: Apresentação da distância geodésica média da rede 145
10.2.20 RF: Geração de redes através dos modelos da literatura 146
10.2.21 RF: Geração de redes de acordo com um novo modelo 150

10.2.22 RF: Importação dos dados de redes pré-definidas 152
10.2.23 RF: Adição de conexões às redes existentes 154
10.2.24 RF: Detecção de caminhos entre duas pessoas 155
10.2.25 RF: Detecção de caminhos entre uma pessoa e várias outras 165
10.2.26 RF: Criação de filtros para visualização dos caminhos 167
10.2.27 RF: Salvamento da imagem da rede gerada 170
10.2.28 RF: Deleção da base de dados 170
10.2.29 RNF: Modularidade 170
10.2.30 RNF: Simplicidade 171
10.2.31 RNF: Paralelismo 171
10.2.32 RNF: Eficiência 171
10.2.33 RNF: Confiabilidade 171
10.2.34 RNF: Portabilidade 171
10.2.35 RNF: Manutenibilidade 171
10.2.36 RNF: Usabilidade 171

11 Considerações Finais **172**
11.0.37 Contribuições 173
11.0.38 Limitações e Trabalhos Futuros 174

Referências Bibliográficas **176**

Lista de Figuras

1.1 Metodologia da Dissertação. 25

2.1 O Evento. 29
2.2 Grafo do Evento. 29
2.3 Jefferson High School. 31
2.4 Colorado Springs. 32
2.5 Amostra de Usuários do Facebook. 34
2.6 Rede Social Hipotética. 35
2.7 Rede Social Real. 36
2.8 Gráfico de Conexões Aleatórias. 38
2.9 Triângulo Social. 42
2.10 Buraco Estrutural. 42
2.11 Pontes Sociais. 43
2.12 A Sociedade de Granovetter. 43
2.13 Modelo Alfa. 45
2.14 Comprimento dos Caminhos no Modelo Alfa. 46
2.15 Coeficiente de Aglomeração. 47
2.16 Modelo Beta. 47
2.17 Impacto das Religações Aleatórias no Modelo Beta. 48
2.18 Distribuição Normal. 50
2.19 Distribuição por Lei de Potência. 51
2.20 Mapas Rodoviários e de Linhas Aéreas dos EUA. 52
2.21 Modelo Sem Escala. 54
2.22 Rede Gerada pelo Modelo Sem Escala. 54
2.23 Desigualdade Triangular. 56
2.24 Multidimensões. 57
2.25 Rede Bipartida. 58

3.1 Grafo da Distância Entre Oito Locais de uma Cidade. 61
3.2 Relaxamento da Aresta (u,v). 63
3.3 Heap Mínimo. 65
3.4 Inicialização do Algoritmo de Dijkstra. 65
3.5 Iterações do Algoritmo de Dijkstra. 65
3.6 Experimento da Ponte Binária. 66
3.7 Obstáculo Sendo Superado Pelas Formigas. 67
3.8 Processo Forrageador das Formigas. 67
3.9 Representação de um Cromossomo. 74
3.10 Node Based Crossover (NBX). 75
3.11 Processo de Reparação do Cromossomo. 76

3.12 Processo de Mutação do Cromossomo. 77

4.1 Arquitetura de um Agente. 79
4.2 Perspectiva Funcional de um Agente. 79
4.3 Hierarquia de Agentes. 80
4.4 Tipos de Ambientes. 81
4.5 Neurônios do Cérebro Humano. 83

5.1 Detectando Objeto Dessemelhante. 85
5.2 Melhoria da Percepção Humana. 85
5.3 Pontes de Königsberg. 86
5.4 Modelo de Referência. 88
5.5 Transformações de Dados. 88
5.6 Símbolos Gráficos Representacionais. 88
5.7 Mapeamento Visual. 89
5.8 Transformações Visuais. 89

6.1 Arquitetura do JADE. 93
6.2 Ciclo de Vida dos Agentes. 94

7.1 Arquitetura do JUNG. 99
7.2 Grafo Gerado pelo JUNG. 101
7.3 Algoritmo EdgeBetweennessClusterer. 102
7.4 Algoritmo DijkstraShortestPath. 103
7.5 Grafo Inicial e Estabilizado. 104
7.6 Grafo Ampliado. 104
7.7 Grafo Rotacionado e Distorcido. 104

8.1 The Oracle of Kevin Bacon. 106
8.2 Vizster. 108
8.3 SocNetV. 109
8.4 Facebook Visualiser, TagGraph, Mentionmapp. 110

9.1 Modelo Gama. 117
9.2 Diagrama de Casos de Uso. 122
9.3 Modelo Conceitual da Aplicação Fluzz. 123
9.4 Modelo Relacional da Aplicação Fluzz. 123
9.5 Arquitetura da Aplicação Fluzz. 124
9.6 Primeiro Elemento do Modelo BestWay. 126
9.7 Segundo Elemento do Modelo BestWay. 127
9.8 Terceiro Elemento do Modelo BestWay. 127
9.9 Representação do Modelo BestWay. 127

10.1 Arquitetura MVC da Aplicação Fluzz. 134
10.2 Layout da Aplicação Fluzz. 135
10.3 Seleção dos Componentes do Grafo. 138
10.4 Movimentação dos Componentes do Grafo. 139
10.5 Mecanismos de Zoom do Grafo. 140
10.6 Rotação e Deformação do Grafo. 140

10.7 Layouts da Aplicação Fluzz. 141
10.8 Visualização do Peso das Conexões. 143
10.9 Visualização do Perfil de Vértices e Conexões. 144
10.10 Modelos de Geração de Redes. 147
10.11 Processo de Adição de Conexões. 155
10.12 Processo de Busca. 163
10.13 Processo de Filtragem de Soluções. 167

11.1 Áreas Envolvidas no Projeto da Aplicação Fluzz. 172

Lista de Tabelas

2.1 Mundos Pequenos 49

9.1 Avaliação dos Modelos de Geração de Redes 116
9.2 Apresentações em Sociedade 119
9.3 Mapeamento de Requisitos Funcionais e Casos de Uso 122
9.4 Dicionário de Dados do Modelo Conceitual 123
9.5 Dicionário de Dados do Modelo Relacional 124

10.1 Configuração de Hardware 134
10.2 Valores Obtidos Pelos Modelos de Rede 150
10.3 Avaliação dos Modelos de Geração de Redes 152
10.4 Performance dos Heaps Perante as Listas 159
10.5 Performance dos Agentes de Pesquisa 164

Lista de Algoritmos

3.1 $Dijkstra(G,w,s,t)$ 64
3.2 $ACO(G,w,s,t)$ 71
3.3 $GA(G,w,s,t)$ 77

Lista de Códigos de Programas

10.1 Inicialização da Aplicação Fluzz 136
10.2 Criação do Banco e das Tabelas do Fluzz no PostgreSQL 136
10.3 Visualizando Grafos no Fluzz 137
10.4 Coloração dos Vértices do Grafo 138
10.5 Alterando *Layouts* de Visualização das Redes 142
10.6 Cálculo do Peso das Arestas 142
10.7 Escondendo os Rótulos das Arestas 143
10.8 Apresentando os Rótulos das Arestas 144
10.9 Cálculo do Coeficiente de Aglomeração 144
10.10 Cálculo da Distância Geodésica Média 146
10.11 Geração de Redes 149
10.12 Instrução para Geração de Redes de Pequena Escala 152
10.13 Montando o Grafo 153
10.14 Adicionando Conexões às Redes 154
10.15 Iniciando Agentes de Pesquisa 156
10.16 Escutando e Encerrando os Agentes de Pesquisa 156
10.17 Marcando o Menor Caminho no Grafo 157
10.18 agenteDijkstra 158
10.19 agenteACS e agenteAS 160
10.20 agenteGA 162
10.21 Definindo Alvos Para Pesquisa 166
10.22 Criando Predicados aos Vértices 169
10.23 Aplicando Filtros às Redes 169
10.24 Criando Subgrafos 170

CAPÍTULO 1

Introdução

Se a época atual da história humana tivesse de ser caracterizada de forma simples, sua descrição poderia ser feita como o período mais altamente, globalmente e inesperadamente conectado já notificado [106]. Para que essa era da conectividade pudesse ser entendida, pesquisadores de diversas áreas como matemática, física, biologia, sociologia e Ciência da Computação uniram esforços para descrevê-la cientificamente, através do estudo intensivo das redes.

A análise de redes sociais sempre foi objeto de interesse sociológico, especialmente a partir de meados do século XX. Todavia, estudos mais aprofundados da dinâmica de redes recentemente foram desenvolvidos, para procurar alternativas às tratativas clássicas que consideravam a formação de redes um processo puramente randômico [23]. Tais estudos passaram a se utilizar da evolução da Teoria dos Grafos com abordagem de dinâmica de sistemas e uso intensivo de recursos computacionais.

Esse alicerce transdisciplinar desafiou a comunidade científica a superar os compartimentos estanques do reducionismo e a explorar, conexão por conexão, a revolução científica adveniente: a nova Ciência das Redes [14]. Proposta originalmente por Duncan Watts, Steven Strogatz e Albert Barabási, a ideia preeminente prescreveu que um padrão oculto está por trás do modo como as pessoas interagem. Os estudos evidenciaram que apesar do crescimento populacional, a distância entre os indivíduos está decrescendo, deixando-os em lados opostos do mundo conectados apenas por seis graus de separação.

Inúmeras revelações referentes a essa ciência inserem-se particularmente no âmbito das redes sociais, mas foram evidenciadas similarmente nas demais redes pesquisadas pelos cientistas. Consequentemente, as pesquisas sobre redes sociais contribuíram com o estudo sobre redes de forma geral, que passaram a ser vistas como parte integral de um sistema em evolução e autoconstituição contínua, e não mais como objetos de estrutura pura, cujas propriedades são fixas no tempo.

Acompanhando os cientistas, que se interessaram em explicar as redes, o início do século XXI caracterizou-se pelo crescente interesse demonstrado pela população mundial sobre o assunto. Isso basicamente se deve ao surgimento da Internet juntamente com a *Web* nos lares, que possibilitou às pessoas uma noção de como as coisas podem

estar interconectadas.

Os indivíduos em sociedade começaram a perceber que estão tão conectados quanto seus computadores, e que, de certa forma, o mundo está encolhendo. Encolhendo, não no sentindo físico ou demográfico, e sim social, reforçando sem precedentes a condição de que a humanidade realmente vive em um mundo pequeno. Estas conexões tornaram-se tão explícitas que hoje quase todas as pessoas estão familiarizadas com as redes sociais *online*, como o *Facebook* [112], que recentemente atingiu a marca recorde de 1.000.000.000 de usuários ativos [1].

As interações, estimuladas e suportadas por novas tecnologias, criam novos fenômenos sociais que transcendem a experiência individual, provocando implicações significativas para o bem coletivo [30]. As redes sociais possibilitam que a humanidade se torne maior do que a soma de suas partes, e a criação de novos modelos de conexão promete contribuir para elevar a interconexão humana a um nível jamais experimentado.

O recurso distintivo das redes sociais *online* é que elas tornam a rede de conexões dos indivíduos visível aos usuários. Além disso, por sua própria definição, as redes sociais *online* são organizadas em torno de pessoas, divergindo-se de outros tipos de grupos *online* ou comunidades, como as *wikis*, que se organizam com base em tópicos.

Bilhões de pessoas têm integrado o uso de redes sociais *online* em seu cotidiano. Elas procuram por antigas amizades, fazem novas conexões, postam *links* favoritos, percorrem as redes de amizades de seus amigos. Compartilhar informações nestes ambientes, compartilhar descobertas importantes, são benefícios que ajudam as pessoas a agirem de forma rápida e eficiente em um mundo dinâmico de transformações frequentes [30]. A cooperação envolvida neste processo gera sinergia, permitindo que a produção humana cresça de forma exponencial.

As redes sociais *online* facilitam a localização e a interação com um enorme número de potenciais amigos, com alto nível de especificidade, além de viabilizar o encontro de pessoas em diferentes domínios. Os indivíduos pertencentes a estes ambientes engendram padrões em que a abertura, conectividade e cooperação, deslocam os limites do impossível ou até do inimaginável, para a região do possível adjacente.

Dentro desse contexto, compreende-se hoje, que a resposta para o sucesso das pessoas em sociedade, encontra-se tanto na estrutura das redes em que operam quanto na possibilidade das mesmas serem navegadas. De muitas formas, os trabalhos realizados pelo matemático Leonard Euler que originaram a Teoria dos Grafos, simbolizam uma importante mensagem desta obra: a construção e a análise de grafos ou redes podem ser determinantes para a compreensão do complexo mundo que rodeia a humanidade.

É neste momento que a Ciência das Redes toma forma, perfazendo um caminho de pesquisas para compreender como mudanças na topologia, afetando somente alguns nós ou *links*, podem abrir portas ocultas, permitindo a emergência de novas possibilidades.

1.1 Motivação

O cérebro humano alcançou sua evolução atual auto-organizando-se a partir de uma forma primitiva de combinação de padrões. Os seres humanos são muito mais hábeis em reconhecer padrões do que em pensar através de combinações lógicas, uma vez que este tipo de reconhecimento abrange a maior parte do circuito neural humano. Segundo Steven Johnson, essas faculdades compensam a velocidade extremamente baixa dos neurônios, que são milhões de vezes mais lentos, por exemplo, que os computadores para resolver problemas de maneira serial [60].

Não obstante, por ser um sistema paralelo de grande porte, com 100 bilhões de neurônios trabalhando simultaneamente, o cérebro pode protagonizar proezas admiráveis de reconhecimento de padrões, como a identificação ou interpretação de imagens [60]. Com esta capacidade, as pessoas podem classificar situações como distantes ou similares, relevantes ou irrelevantes, ou até mesmo encaixar pequenas pistas, formando uma imagem bem definida de dados muitas vezes difíceis de serem interpretados pelo computador.

Dessa forma, se um mapeamento estrutural gráfico das redes sociais pudesse ser disponibilizado, o cérebro humano utilizaria de sua principal habilidade cognitiva para prover informações, muitas vezes despercebidas pelas melhores máquinas.

Em um primeiro momento, as tecnologias capazes de representar redes sociais, como as redes sociais *online*, surgiram para mostrar o interesse intrínseco do ser humano em se conhecer e conhecer seus amigos em rede [30]. Em um ambiente capaz de representar graficamente as redes sociais, em vez de apenas conhecerem quem são seus amigos e, possivelmente, os amigos de seus amigos, os indivíduos poderiam olhar além de seus horizontes sociais e visualizar seu espaço em uma vasta rede social mundial. Atualmente são raras as aplicações que disponibilizam esse tipo de recurso, o que prejudica uma análise aprimorada e instintiva das pessoas.

Por outro lado, a utilização da alta velocidade de processamento serial dos computadores não poderia ser negligenciada para a realização de outro importante procedimento nas redes sociais: o processo de busca. Segundo o físico Mark Buchanan, a concepção de uma sociedade em rede foi evidenciada na década de 1980, quando os indivíduos perceberam que a melhor estratégia para encontrar informações sobre questões abstrusas, era explorar os tentáculos de sua rede social, na busca de prováveis colaboradores. Para o pesquisador, "as pessoas estendiam antenas a amigos e conhecidos, e esperavam que, em algum ponto da cadeia, alguma preciosidade aparecesse em seu caminho. Um amigo do vizinho do tio poderia ser a pessoa-chave procurada por alguém" [23].

Encontrar determinada pessoa, que supostamente poderia disponibilizar uma informação procurada, tornou-se ainda mais importante atualmente, quando problemas têm de ser resolvidos às pressas e ninguém tem uma ideia clara de como resolvê-los. Não

obstante, definir qual pessoa procurar, e qual o melhor caminho percorrer para encontrá-la em um emaranhado de nós e conexões, invariavelmente confundiria e estressaria até a mais vigorosa mente que se aventurasse sobre tal domínio.

Isso ocorre, pois o cerne da interconectividade humana não está na simples estrutura das díades, ou conexões formadas pelos contatos diretos de uma pessoa. Diferentemente disso, a compreensão fundamental é de que estas díades se agregam para formar teias enormes de laços sociais que vão muito além das conexões diretas, implicando em uma complexidade de ramificação exponencial [30]. Dessa forma, distâncias de dois graus de separação nas redes sociais já tornam o processo de busca extremamente difícil de ser realizado pelos seres humanos, e acima deste nível, tornam-no praticamente inconcebível.

Além disso, um importante fator, certamente é considerado pelos indivíduos ao utilizarem suas redes sociais para realizarem o processo de busca: a influência. Este tipo de prestígio social faz-se presente na maioria das relações de amizade, e pode ser preponderante para a efetivação de uma nova conexão, intermediada por elos comuns. Diversos níveis de influência existem nos distintos caminhos que unem direta ou indiretamente duas pessoas, e algumas variáveis são cruciais para esta diferenciação, como a força dos laços de amizade, e o comprimento dos caminhos. Tais variáveis precisam ser respectivamente maximizadas e minimizadas para que se aumente o fator de influência, e consequentemente a possibilidade de conexão e cooperação em rede.

A dificuldade fundamental constituída em todo o processo de busca social ocorre pela tentativa de resolver um problema global, usando apenas informações locais sobre a rede. Em consequência, é impossível saber qual dos caminhos que partem dos indivíduos chega até a pessoa-alvo com o maior fator de influência. A cada grau de separação, uma nova decisão deve ser tomada, e a escolha por determinado caminho na rede pode inicialmente levar para o que parece ser a direção errada, pois não existe um mapa completo do percurso.

Neste momento, além da disponibilização de um mapeamento gráfico das redes sociais, seria de extrema utilidade a disponibilização de recursos inteligentes capazes de perfazer rapidamente longos e sinuosos caminhos a procura de alvos definidos dentro das redes sociais. Agentes computacionais, que possam através de suas heurísticas e inferências, identificar alvos, classificá-los em ordem de relevância, e reconhecer quais são os melhores caminhos que levam a pessoa-origem à pessoa-destino.

Na próxima geração de redes sociais *online*, portanto, será inaceitável pensar em *sites* que não ofereçam recursos gráficos de visualização da rede e que não permitam a busca inteligente de potenciais parceiros ou colaboradores. Tais mecanismos de busca serão capazes de explorar as conexões sociais existentes nas redes, maximizando a possibilidade de influência entre os indivíduos, e consequentemente elevando o nível de interação e cooperação humana a patamares sem precedentes.

1.2 Objetivos

O principal objetivo deste trabalho é a realização de um estudo científico sobre redes sociais, visando reunir conhecimentos importantes sobre seu processo evolutivo, para a proposição de modelos e ferramentas capazes de atender a três metas específicas:

- Representar as redes sociais graficamente, oferecendo diversos recursos para visualização e interação com as mesmas;
- Simular a geração das redes sociais de acordo com os principais modelos propostos na literatura, e conforme um novo modelo concebido nesta obra;
- Disponibilizar mecanismos de busca para a localização de pessoas específicas nas redes sociais, alicerçados em modelos que possam maximizar a probabilidade de influência entre os indivíduos.

A definição dessas metas alicerçou-se em um intuito social maior, de promover interações e futuras conexões entre os integrantes das redes analisadas, e para que elas pudessem ser atendidas, técnicas computacionais e modelos conceituais foram desenvolvidos e implementados em uma aplicação prática. Tal aplicação foi também responsável pelo processo de verificação e validação de todo o arcabouço teórico proposto, uma vez que os recursos de simulação disponibilizados pela mesma produziram dados passíveis de análise exploratória e confirmatória de evidências.

A aplicação foi denominada Fluzz, em homenagem ao livro do físico, escritor, sociólogo e um dos principais estudiosos sobre redes no Brasil, Augusto de Franco. O livro de Franco intitulado, "Fluzz: Vida humana e convivência social nos novos mundos altamente conectados do terceiro milênio" [46], realiza uma abordagem sociológica sobre os diversos benefícios de convivência em rede.

Em uma das principais passagens do livro, Franco menciona:

> O mundo das redes não é um mundo: é um multiverso de interações. Não existe uma mesma realidade para todos: são muitos os mundos. Tudo depende das fluições em que cada um se move, dos emaranhamentos que se tramam, das configurações de interações que se constelam e se desfazem, intermitentemente. Quanto mais distribuída for a topologia de uma rede, mais-fluzz ela será. Quer dizer, mais interatividade haverá. Conhecer as redes é interpretar modos-de-interagir (reconhecendo padrões). O que só se pode conseguir interagindo (estabelecendo conexões).

É expressiva e contundente a mensagem transmitida por Franco durante toda a obra citada. Reconhecer padrões e estabelecer conexões refletem inextrincavelmente a tônica principal enredada pela aplicação desenvolvida neste trabalho, que empresta orgulhosamente o título de seu livro.

Para contribuir com o aumento da atividade conectiva e interativa das pessoas em rede, a aplicação utiliza-se, por exemplo, de modelos para a avaliação dos laços de amizade construídos. O emprego de tais modelos mostra-se como premissa essencial dentro desta conjectura, pois viabiliza a possibilidade de analisar a distância, e consequentemente o nível de proximidade existente entre os indivíduos, criando condições para a realização de medições relevantes da probabilidade de influência recíproca.

A aplicação foi construída sobre os moldes dos sistemas multiagentes, capazes de utilizar as habilidades de diferentes agentes de *software*, que interagem na resolução de problemas. Este tipo de sistema pode apresentar maior rapidez na realização de tarefas, devido ao paralelismo obtido [12]. A modularidade garantida com os sistemas multiagentes é outra característica relevante, pois quando estes não conseguem resolver determinada tarefa, geralmente, é mais simples projetar e incluir um novo agente do que substituir o sistema inteiro.

Além do objetivo principal apresentado, esta obra pretende alcançar os seguintes objetivos secundários que concorrem ao mesmo ponto prevalecente:

- Disseminar a compreensão dos conceitos e linguagem da Ciência das Redes, objetivando suas utilizações no tratamento de problemas complexos em rede;
- Expor os principais fundamentos sobre Sistemas Multiagentes, indicando as principais tecnologias utilizadas nesta subárea da Inteligência Artificial, e apresentando os pontos diferenciais da ferramenta escolhida no trabalho;
- Utilizar a aplicação desenvolvida para inferir informações referentes aos dados processados;
- Exibir os conceitos e indicar tecnologias capazes de trabalhar com visualização gráfica de dados, apresentando as vantagens da ferramenta utilizada na aplicação;
- Esclarecer e contextualizar o critério utilizado no trabalho para definir a força dos laços sociais entre os integrantes das redes;
- Introduzir o modelo de rede criado para viabilizar a avaliação qualitativa dos diversos caminhos que conectam duas pessoas específicas nas redes sociais;
- Apresentar a arquitetura projetada para definir as formas de interação entre os agentes de *software* responsáveis pela realização das buscas, bem como seus respectivos modos de convergência;
- Introduzir os conceitos de otimização do problema do caminho mínimo, bem como as características dos algoritmos utilizados na aplicação, como Algoritmo de Dijkstra, Otimização por Colônia de Formigas e Algoritmos Genéticos.

1.3 Metodologia

A criação desta obra foi decomposta em 10 etapas, ilustradas na Figura 1.1.

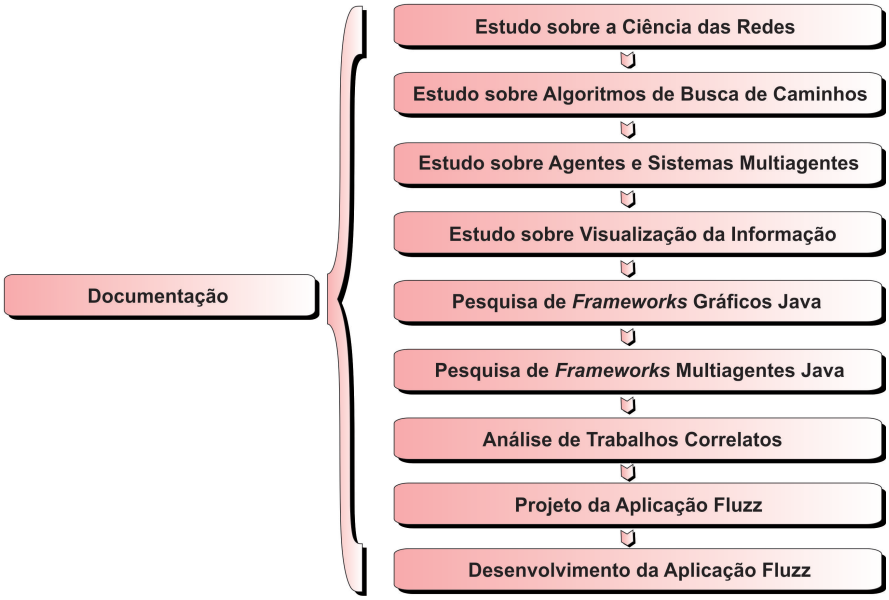


Figura 1.1: *Metodologia da Dissertação.*

Iniciando o processo, a primeira etapa consistiu no estudo teórico necessário para a compreensão dos conceitos erigidos pela Ciência das Redes, o que viabilizou a proposição fundamentada dos modelos sociais criados nesta dissertação.

A próxima etapa caracterizou-se por uma exploração realizada na área de algoritmos de busca de caminhos, objetivando comparar algumas técnicas de busca propostas na literatura, que pudessem ser utilizadas para a criação de um mecanismo de busca social.

Por conseguinte, iniciou-se um processo de estudo do paradigma de programação orientado à agentes, abordando sua múltipla utilização em Sistemas Multiagentes.

A ciência Visualização da Informação recebeu o foco do estudo em sequência, sendo avaliadas as técnicas e orientações referentes à representação gráfica de dados.

Finalizado o estudo teórico, foi iniciada uma pesquisa sobre *frameworks* construídos na plataforma Java. O objetivo era encontrar uma ferramenta de código aberto que possibilitasse a criação de sistemas sob o paradigma multiagente, simplificando e facilitando este desenvolvimento. O *framework JADE* (*Java Agent DEvelopment framework*) foi adotado devido a sua adequação à maioria dos pré-requitos estipulados para uma arquitetura de comunicação flexível e eficiente, além de atender a todas as especificações da principal organização padronizadora de tecnologias baseadas em agentes.

Na etapa subsequente, uma nova pesquisa sobre *frameworks* Java entrou em vigor. Desta vez, a intenção foi de analisar as ferramentas disponibilizadas na academia para criação e manipulação de componentes gráficos, mais especificamente que reunissem condições de trabalhar com grafos. A ferramenta *JUNG* (*Java Universal Network/Graph*

Framework) apresentou as melhores características dentre todas as observadas, sendo então a designada para a construção da aplicação.

Ao iniciar a etapa de trabalhos relacionados, foram levantados alguns projetos com objetivos semelhantes ao desta pesquisa, ou que de alguma forma estavam, direta ou indiretamente relacionados com o tema. A ideia preeminente foi conhecer projetos e ferramentas que pudessem colaborar com os propósitos essenciais da aplicação a ser desenvolvida, através de técnicas ou modelos pré-concebidos.

Em sequência foi iniciado o projeto da aplicação Fluzz, pelo qual foram identificados e concebidos os requisitos e os modelos a serem implementados, visando atender integralmente os objetivos desta obra.

Finalizando todo o processo, a última etapa consistiu no desenvolvimento efetivo da aplicação Fluzz, com a estipulação congruente das tecnologias e processos adotados.

A documentação, fundamentada essencialmente na escrita da dissertação, relatórios e artigos, foi realizada concomitantemente no decorrer de todas as outras etapas.

1.4 Organização da Dissertação

Este trabalho encontra-se organizado em 10 capítulos, além desta introdução. No Capítulo 2, são apresentados os principais conceitos e estudos realizados na área de Ciência das Redes, através de uma abordagem interdisciplinar.

No Capítulo 3, são apresentados alguns algoritmos estudados, detalhando suas respectivas lógicas ou heurísticas utilizadas na busca otimizada de caminhos.

O Capítulo 4 destina-se à discussão sobre Agentes e Sistemas Multiagentes, considerando as principais vantagens de suas utilizações.

O Capítulo 5 aborda a temática envolvida com a Visualização de Informação, analisando como construir representações gráficas expressivas e eficientes.

No Capítulo 6, são introduzidos alguns dos principais *frameworks* Java para construção de Sistemas Multiagentes, justificando a escolha por determinado *framework*.

O Capítulo 7 informa alguns dos principais *frameworks* Java para construção de aplicações gráficas, também justificando a opção pelo *framework* utilizado no trabalho.

No Capítulo 8, são descritos os principais trabalhos relacionados com a aplicação desenvolvida, destacando-se: *The Oracle of Kevin Bacon*, *Prefuse* e *SocNetV*.

O Capítulo 9 apresenta os requisitos funcionais e não funcionais, além dos modelos que serviram como base para o projeto da aplicação Fluzz.

O Capítulo 10 é dedicado ao desenvolvimento da aplicação Fluzz, utilizando as especificações do capítulo anterior.

Finalmente, o Capítulo 11 fornece algumas considerações finais, enumerando contribuições e possíveis trabalhos futuros.

CAPÍTULO 2

Ciência das Redes

Um dos aspectos mais fascinantes do surgimento de uma nova ciência é a linguagem particular que ela engendra, permitindo discorrer sobre ideias e questões complexas de serem descritas antes de sua emergência. Ao construir uma linguagem para expor sobre redes, que seja precisa o bastante para descrever não apenas o que são, mas também que tipos de diferentes redes existem no mundo, a Ciência das Redes está fornecendo ao conceito uma força analítica real.

O pensamento de redes está em via de invadir todos os domínios da atividade humana e a maioria dos campos de investigação [14]. É mais do que uma simples perspectiva. As redes são, por sua própria natureza, a urdidura dos sistemas complexos, e os nós e *links* impregnam profundamente todas as estratégias voltadas para a abordagem do universo interconectado [106].

Proposta inicialmente pelos pesquisadores Duncan Watts, Steven Strogatz e Albert Barabási a Ciência das Redes adota uma nova visão da sociedade orientada por redes, além de tentar compreender as consequências da interconectividade. Neste contexto, os últimos anos vêm propiciando uma explosão de pesquisas e interesse mundial em busca de um novo paradigma com o qual se possa descrever, explicar e, em última análise, compreender a era da conectividade.

O propósito essencial desta ciência consiste em ajudar os indivíduos a entender, tanto a estrutura dos sistemas conectados quanto as formas pelas quais diferentes tipos de influência se propagam através deles [106]. Obviamente, reconhecer esses padrões e compreender suas origens é apenas um primeiro passo. Será também necessário saber como influenciá-los, e como utilizar as propriedades das redes a favor dos seres humanos.

2.1 Complexidade e Emergência

O reducionismo foi a força propulsora de boa parte da pesquisa científica do século XX. Para compreender a natureza, segundo seus preceitos, é necessário inicialmente que seus componentes sejam decifrados. A premissa é que uma vez compreendida as partes, consequentemente aprende-se o todo [23]. Naturalmente, durante décadas os cien-

tistas viram o mundo por suas partes constitutivas. Foram instruídos a estudar os átomos para entender o universo, as moléculas para compreender a vida, os genes individuais para entender o complexo comportamento humano.

Atualmente, os cientistas adquiriram um grande conhecimento sobre as partes, porém ainda estão longe de entender a natureza como um todo, já que a remontagem revelou-se muito mais difícil do que o previsto. Para Buchanan a razão é simples: "sob a égide do reducionismo, logo se depararam com a dura muralha da complexidade" [23].

Os cientistas entenderam que a natureza não é um quebra-cabeça bem projetado, que apresenta uma única forma de montagem. Em sistemas complexos, cujas propriedades não são uma consequência natural de seus elementos constituintes, os componentes podem articular-se de modos tão diversos que os pesquisadores levariam bilhões de anos para tentar todos. Não obstante, a natureza reúne as partes com uma precisão aguçada ao longo de toda sua história [106]. Ela o faz explorando as leis gerais da auto-organização, cujas raízes, em ampla medida, ainda constituem um mistério para quem as estudam [60].

O recente sequenciamento do genoma humano revelou que o código básico de toda a vida humana consiste em cerca de apenas trinta mil genes, muito menos do que os cientistas haviam imaginado, por volta de cem mil [103]. Isso é particularmente intrigante, já que algumas plantas têm cerca de vinte e cinco mil genes. Consequentemente, como afirma Buchanan, "ou as pessoas não são tão sofisticadas quanto imaginam ou não é apenas o número de genes que determina a complexidade de um organismo" [23].

Logo, a explicação de toda a complexidade da biologia humana, claramente não provém da complexidade dos elementos individuais do genoma, que se totalizam em um número pouco maior do que o encontrado no mais singelo dos organismos. Ao contrário, ela provavelmente deriva do fato de que traços genéticos são raramente expressados por genes isolados. Embora os genes, como as pessoas, existam como unidades individuais, eles funcionam por interação, e os padrões de interação correspondentes podem exibir complexidade quase ilimitada.

Um cérebro humano, por exemplo, é em certo sentido, bilhões de neurônios conectados em uma grande massa eletroquímica. Mas, na realidade, ele claramente é bem mais do que isso, com propriedades como consciência, memória e personalidade, cuja natureza não pode ser simplificada como um agregado de neurônios. Essa propriedade relativa aos sistemas complexos, de um padrão organizacional não planejado que pode emergir devido à auto-organização de seus componentes, é conhecida como *emergência*.

Mais do que conhecer as partes, é necessário compreender a dinâmica organizacional por trás da formação da rede, que faz emergir sua configuração em determinado momento, transcendendo assim, o reducionismo. Entender as inter-relações simultâneas e seus resultados não planejados é essencial para tratar os problemas complexos em rede, e, portanto, torna-se o alicerce diferencial da nova ciência apresentada.

2.2 Grafos ou Redes

Reduzida ao seu esqueleto básico, uma rede constitui simplesmente um conjunto de objetos conectados entre si de certo modo. Roteadores na Internet ou neurônios em um cérebro são exemplos de redes, mas que são completamente distintas de alguma forma.

Atualmente, o ramo da matemática conhecido como Teoria dos Grafos, proposto originalmente pelo matemático Leonhard Euler, constitui a base do pensamento acerca de redes. Séculos depois de Euler, ele se tornou um campo perfeitamente desenvolvido, que tem recebido a contribuição de grandes pesquisadores.

Para exemplificar a formação de uma rede, o físico Albert Barabási criou a ilustração de um evento com dez convidados como representado na Figura 2.1 [14], em que, a princípio, nenhum deles se conhecem. Os laços, ou vínculos sociais, são formados à medida que os convidados começam a conversar em pequenos grupos.

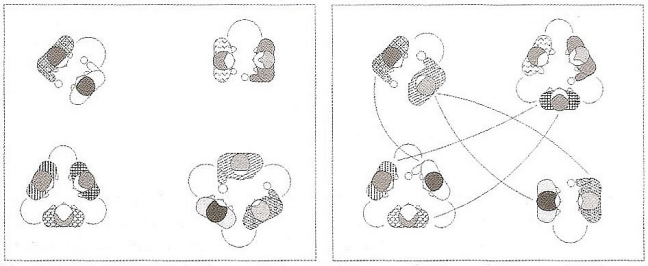


Figura 2.1: *O Evento.*

No início, os grupos estão isolados uns dos outros, como no painel esquerdo. Com o passar do tempo, três convidados migram para diferentes grupos, emergindo um grande agrupamento, ilustrado no painel direito. Embora nem todos se conheçam, existe agora uma única rede social constituída por todos os convidados. É possível encontrar um caminho por entre dois indivíduos quaisquer, simplesmente seguindo os vínculos sociais.

Os convidados são os nós, e cada encontro cria um elo social entre eles. É assim que surge uma rede de conhecidos, ou um grafo, configurando um aglomerado, ou um *cluster*, de nós conectados por *links*. Independente da identidade e da natureza dos nós e dos *links*, matematicamente eles formam uma mesma estrutura: um grafo ou uma rede [14]. A representação em forma de grafo da Figura 2.1 pode ser visualizada na Figura 2.2.

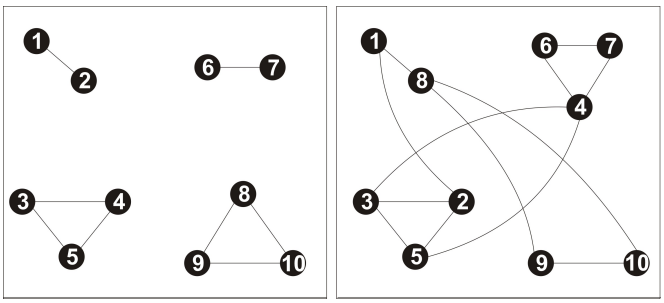


Figura 2.2: *Grafo do Evento.*

Inicialmente não há nenhuma novidade nessa explanação. O cerne da questão, porém, é que no passado, redes foram vistas como objetos de estrutura pura, cujas propriedades são fixas no tempo, o que se distancia da realidade. As redes reais representam populações de componentes que estão em atividade constante, gerando energia, e a estrutura das relações entre estes componentes afeta o comportamento individual, e consequentemente o comportamento do sistema como um todo.

Além disso, redes são objetos dinâmicos não apenas porque coisas acontecem em seu interior, mas porque elas próprias evoluem no tempo, impelidas pelas ações de seus componentes. Na era da conectividade, portanto, o que acontece e o modo como acontece depende da rede, que por sua vez, depende do que havia acontecido antes. É esta nova visão de rede que é verdadeiramente nova na Ciência das Redes [106].

2.3 Redes Sociais

Em um sentido básico, uma rede social é um conjunto de dois tipos de elementos: seres humanos e suas conexões entre si [87]. A organização destas redes não costuma ser imposta por algum mecanismo, evoluindo organicamente da tendência natural de cada pessoa buscar e fazer suas relações conjugais, profissionais, e sociais de maneira geral.

Existem duas características fundamentais que regem este tipo de rede, sejam estas pequenas comunidades ou a rede social mundial. Primeiro, há uma conexão, que indica quem está conectado a quem [30]. Quando um grupo é constituído como uma rede, há um padrão específico de *links* que conectam os componentes envolvidos, descrevendo como os mesmos estão interligados: a *topologia*. Em redes sociais, estas conexões são constituídas pelos complexos laços de amizade criados pelos seres humanos, e o modo como a rede é construída ou visualizada depende de como estes laços são definidos.

Segundo, há contágio, que diz respeito ao que flui ao longo dos laços sociais. Poderiam ser informações, ou vírus. Cada um desses fluxos se comporta de acordo com suas próprias regras [30]. Entender por que as redes sociais existem e como elas funcionam, exige que as regras em relação à conexão e ao contágio sejam entendidas. Esses princípios explicam como os laços fazem das redes sociais verdadeiros sistemas complexos, transformando-as em algo bem maior do que uma mera coleção de indivíduos.

Em um estudo experimental, que agrupava um conjunto de dados contendo indicadores de emoções e conexões sociais ao longo do tempo, os pesquisadores Nicholas Christakis e James Fowler mapearam os grafos da felicidade, da obesidade, dos fumantes, dentre outros, de uma amostra de 12.067 pessoas de Framingham, Massachussets. Eles estavam especialmente interessados em determinar se a disseminação das emoções ocorria além das amizades diretas, e por intermédio deste estudo, definiram as regras fundamentais, que regulamentam tanto a formação quanto a operação das redes sociais [30]:

- **Regra 1: As pessoas modelam suas redes.** As pessoas definem a estrutura de suas redes de três modos: decidem com quantas pessoas desejam se conectar, influenciam a densidade de interconexão de seu amigos, servindo como mediadores, e controlam quem está no centro da rede através da dinâmica de encontros;
- **Regra 2: As redes sociais modelam as pessoas.** Quando as pessoas com as quais um indivíduo está conectado tornam-se mais bem conectadas, isso reduz a distância que o mesmo tem de percorrer para alcançar todos na rede. O indivíduo se torna mais central, o que o deixa mais suscetível ao que esteja fluindo pelos laços sociais;
- **Regra 3: Os amigos de uma pessoa a afetam diretamente.** Um determinante fundamental do fluxo é a tendência dos seres humanos se influenciarem. Cada um dos laços diretos oferece oportunidades para influenciar e ser influenciado;
- **Regra 4: Os amigos dos amigos de uma pessoa a afetam indiretamente.** Os amigos podem influenciar as pessoas, mas os amigos deles também podem influenciá-las. Isso é uma ilustração da *disseminação hiperdiádica*, ou tendência de os efeitos se disseminarem para além das conexões sociais diretas;
- **Regra 5: A rede tem vida própria.** As redes sociais possuem propriedades e funções que não são controladas ou percebidas pelas pessoas em seu interior, sendo estas somente entendidas, ao se estudar o grupo inteiro, e não indivíduos isolados.

A relevância do estudo da estrutura de redes sociais consiste no entendimento de como esta estrutura afeta o fluxo. Os sociólogos James Moody, Katherine Stovel e Peter Bearman visualizaram esta dinâmica, ao mapearem redes sociais americanas [17].

Na rede da *Jefferson High School*, apresentada na Figura 2.3 [17], uma pessoa *X* pode ser atingida por um vírus, ou por uma inovação qualquer. A pessoa *A* está a cinco passos distantes da pessoa *X*, o que possibilita ao vírus em cinco saltos alcançá-la.

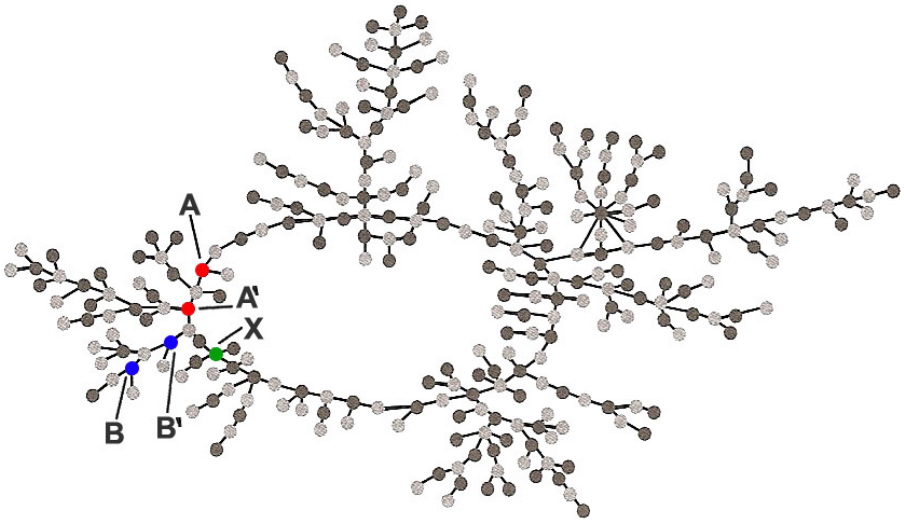


Figura 2.3: *Jefferson High School.*

Ocasionalmente, se um laço social a dois passos de distância da pessoa *A*, *A'*, for cortado, ela não pode concluir que está evitando a doença, pois a estrutura permite que o vírus circule por um outro caminho maior e, mesmo assim, alcance-a.

Ainda na Figura 2.3, a pessoa *B*, assim como a *A*, também está conectada a três pessoas, distante cinco passos da pessoa *X*. Mas agora, se um laço a duas pessoas distantes da pessoa *B*, *B'*, for cortado, a pessoa *B* estará isolada da epidemia. Sua posição na rede é bastante diferente da pessoa *A*, mas ela não tem a perspectiva para visualizar isso.

Sem essa visão completa da rede, não há como obter tal perspectiva. A pessoa *B* está a mercê da rede em que reside, com algum controle sobre suas conexões diretas, mas sem controle em relação às pessoas com quem está conectada indiretamente [30].

Comparando com outra rede de pessoas em *Colorado Springs*, Figura 2.4 [17], percebe-se que a pessoa *A* também tem três contatos, neste caso mais bem conectados, tornando-a mais suscetível ao que flui na rede. Por essa perspectiva, as pessoas tornam-se vulneráveis ao que circula nas redes não por quem são, mas por quem conhecem [30].

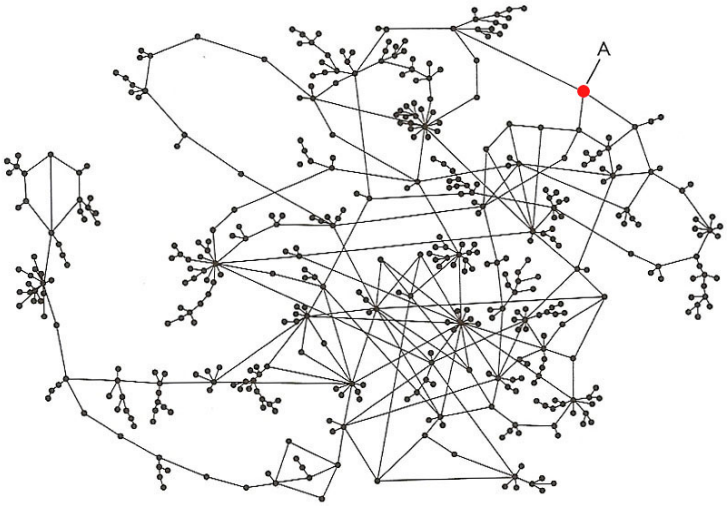


Figura 2.4: *Colorado Springs.*

2.3.1 Redes Sociais *Online*

A invenção de cada novo método de comunicação contribuiu para a extensão e a suplementação dos meios convencionais das pessoas formarem conexões. No caso particular da Internet, as relações surgidas *online* podem ser desimpedidas por várias circunstâncias ou características como geografia, timidez e até discriminação.

As redes sociais *online*, essencialmente as mais populares durante os últimos anos, são *sites* de serviços que permitem aos usuários criarem um perfil pessoal público, ou semipúblico, em um ambiente de acesso limitado [22]. Exibindo uma lista de outros usuários com que cada um compartilha uma conexão, estes *sites* tornam possível a visualização das conexões dos usuários no sistema.

A amplitude e o imediatismo da cultura da rede social *online*, possibilitam muito mais a disseminação de comportamentos. O *feedback* instantâneo obtido pode ser fornecido para um breve impulso, que poderia ter desaparecido sozinho em gerações anteriores que ainda não contavam com essa tecnologia.

O primeiro *site* de rede social reconhecível, *SixDegress.com*, foi lançado em 1997 [30]. Mesmo atraindo muitos usuários, o *site* falhou como negócio em 2000, pois o mercado ainda não estava pronto para o conceito.

Em 2002, o *Friendster* foi lançado para competir com o *site* de relacionamento *Match.com* [30]. Diferentemente do *Match.com*, que se concentrava em facilitar a apresentação de estranhos, o *Friendster* explorou a ideia de que os amigos de amigos seriam um grupo mais interessante para atrair parceiros românticos.

Além disso, o *Friendster* revelou que saber algo sobre os amigos de alguém poderia dizer mais sobre esta pessoa do que ela estaria preparada para revelar. Este *site* definitivamente mostrou o interesse duradouro de muitas pessoas em saber a quem estão conectadas, e como tais conexões são viabilizadas.

Com a queda de popularidade do *Friendster*, o *MySpace*, lançado em 2003, acabou ganhando grande notoriedade, especialmente entre os artistas. Não obstante, ele foi logo obscurecido por um forte concorrente: o *Facebook* [30].

A rede social *online Facebook* teve início em *Harvard*, em 2004, mas sua história está, de fato, arraigada em um fenômeno real. O nome vem de uma instituição existente há décadas em *Harvard*. Anualmente a instituição publicava e distribuía um livro, *facebook*, que mostrava fotos de todos os estudantes de cada classe e onde viviam no campus [30].

Vinte e cinco anos depois, Mark Zuckerberg colocou o *facebook online*, e isso se tornou tão popular que rapidamente se espalhou para outras instituições. Em Junho de 2008, o *Facebook* ultrapassou mundialmente o *MySpace* em número total de usuários, tornando-se a maior rede social *online* já existente [30].

Segundo Christakis e Fowler, a Internet permite quatro novas características, que são modificações radicais de tipos existentes de interação social [30]:

- **Enormidade:** Vasto aumento na escala das redes pessoais e no número de pessoas que podem ser alcançadas;
- **Vizinhança social:** Expansão da escala da qual as pessoas podem compartilhar informações e contribuir com esforços coletivos;
- **Especificidade:** Aumento impressionante na particularidade dos laços formados;
- **Virtualidade:** Capacidade de assumir identidades visuais.

A congruência dos laços sociais *online* pode ser vista graficamente. A Figura 2.5 ilustra a diferença na capacidade de conexão entre a vida real e a *online* de 140 estudantes

universitários do estudo de Christakis e Fowler sobre o *Facebook* [30]. O que era uma tênue rede tornou-se um novelo entrelaçado de fios.

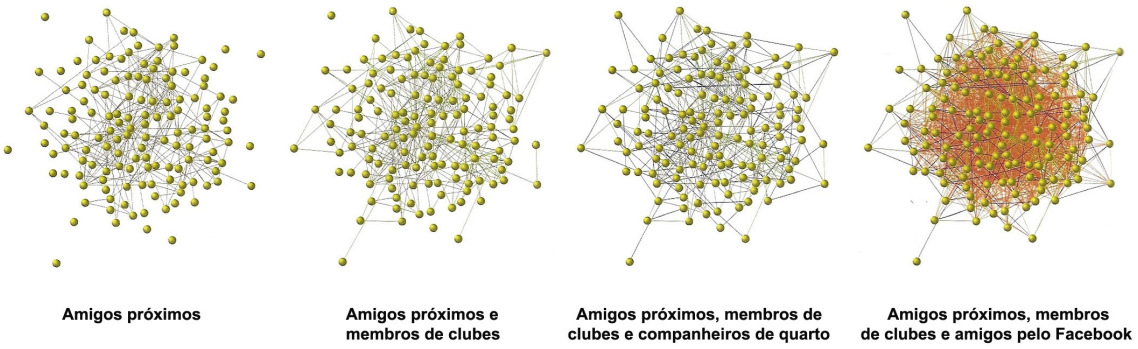


Figura 2.5: *Amostra de Usuários do Facebook.*

A mudança drástica ocorrida, é que no passado, costumava-se exigir muito para a obtenção de qualquer tipo de reforço, ou influência, o que hoje pode depender de apenas um clique. Essa condição oferece um nível de contato humano jamais outrora visto.

Toda essa capacidade de conexão pode contribuir significativamente para a localização de indivíduos específicos nas redes sociais *online*. Não obstante, esse benefício também vem com um preço. Estar mais conectado significa poder localizar mais pessoas, o que torna a tarefa de encontrar a pessoa certa, um exercício de extrema paciência. Em breve, tudo o que se dissemina entre as pessoas se propagará a distâncias maiores e mais rapidamente, solicitando novos recursos que acompanhem a escala de interações.

2.4 Seis Graus de Separação

Em 1967, o psicólogo social Stanley Milgram estava interessado em uma hipótese não resolvida da comunidade sociológica da época [74]. A hipótese afirmava que o mundo, se visualizado como uma enorme rede de relações sociais, de alguma forma era pequeno. O *fenômeno do mundo pequeno*, como era conhecido, garantia que qualquer pessoa pudesse ser contatada através de uma rede de amigos em apenas alguns passos.

Milgram realizou um experimento, cujo objetivo era descobrir quantas pessoas seriam necessárias para conectar dois indivíduos selecionados ao acaso nos Estados Unidos. Ele escolheu duas pessoas-alvo, em Sharon, e em Boston, no estado de Massachusetts. Escolheu Wichita, em Kansas, e Omaha, em Nebraska, como pontos de origem para o estudo, pois estas cidades pareciam bem distantes, tanto geográfica, como socialmente de seus destinos. Na época, havia pouco consenso em torno de quantos *links* seriam necessários para realizar estas conexões, sendo a estimativa típica, na casa de centenas [14].

O experimento de Milgram envolveu a remessa de cartas a moradores aleatoriamente selecionados de Wichita e Omaha pedindo-lhes que participassem de um estudo

sobre contato social, cuja tarefa era tentar contatar uma das pessoas-alvo, através de suas redes de amizades. As cartas continham breve sumário do propósito do estudo, uma foto, o nome e o endereço, bem como outras informações sobre uma das pessoas-alvo.

Caso não conhecesse diretamente a pessoa-alvo, a pessoa-origem deveria optar por alguém em sua rede, que por sua vez pudesse conhecê-la, e assim sucessivamente até que a pessoa-alvo fosse localizada. Para o psicólogo, se a quantidade de *links* oscilasse em torno de uma centena, provavelmente o experimento malograria, pois sempre existem indivíduos ao longo da cadeia que não cooperam.

De forma surpreendente, 43 das 160 cartas chegaram, o que permitiu o cálculo do número médio de pessoas intermediárias, constatado por Milgram pela pequena cifra de 5,5 [14]. Arredondando-se, portanto, chega-se aos famosos *seis graus de separação*, que de fato, representam seis passos ou seis *links* de distância.

Em 2002, os pesquisadores Peter Dodds, Roby Muhamad e Duncan Watts replicaram o experimento de Milgram em uma escala global utilizando o *e-mail* como modo de comunicação. Eles recrutaram milhares de voluntários para enviar uma mensagem a alvos espalhados em todo o mundo. Surpreendentemente, também foram necessários, grosso modo, seis passos em média para enviar o *e-mail* a cada pessoa-alvo, ratificando a estimativa original de Milgram [39].

Os seis graus de separação intrigam por dois motivos. Primeiro, por sugerirem que a sociedade mundial pode ser navegada pelos *links* sociais de uma pessoa a outra. Segundo, porque em uma rede de mais de 7 bilhões de nós, qualquer par de nós encontra-se, em média, a seis *links* um do outro, evidenciando a problemática.

Em uma tentativa de modelar o problema, Watts exemplificou a situação em que determinada pessoa, *Ego*, possui cem amigos [106]. Cada amigo de *Ego* também possui cem amigos. Portanto, a um grau de separação, *Ego* está conectada a cem pessoas, e a dois graus, dez mil. A três graus, chega-se a um milhão. A quatro, cem milhões, e a cinco dez bilhões de pessoas. Desta forma, se todos no mundo tivessem cem amigos, em seis passos seria possível contatar toda a população, raciocínio ilustrado na Figura 2.6 [106].

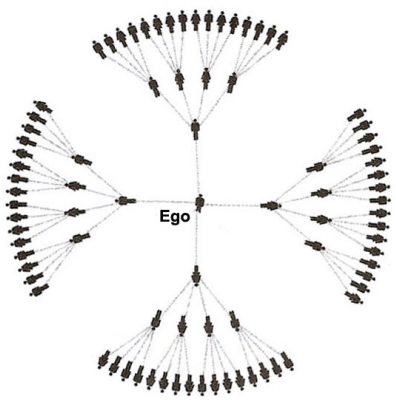


Figura 2.6: *Rede Social Hipotética.*

O mínimo de inclinações sociológicas já permite a detecção do principal erro desse raciocínio. Se uma pessoa pensar nos seus melhores amigos, e em sequência, se lembrar dos melhores amigos destes últimos, certamente ela verificará a sobreposição de indivíduos neste círculo de amizade. Essa observação se revelou como uma característica quase universal, não apenas de redes sociais, mas de redes em geral. Elas exibem o que se chama de aglomeração, ou *clustering*, o que equivale a dizer que a maioria dos amigos de alguém também tem, em algum nível, amizade entre si, criando redundância [106].

De fato, como observou Watts, redes sociais lembram muito mais a Figura 2.7 [106]. Isso ocorre, pois as pessoas tendem a ter grupos de amigos, sendo cada grupo um pequeno aglomerado baseado em experiências, localização ou interesses compartilhados.

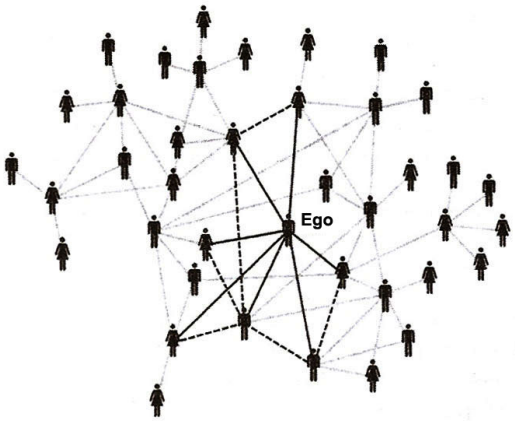


Figura 2.7: *Rede Social Real.*

O paradoxo das redes sociais que o experimento de Milgram ressaltou, é que apesar do mundo ser altamente aglomerado, as pessoas ainda podem contatar qualquer indivíduo em uma média de poucos passos. Embora a hipótese do mundo pequeno tenha permanecido incontestada por mais de três décadas, continua surpreendente até hoje.

2.5 Três Graus de Influência

O fato das pessoas estarem conectadas a seis graus de separação, não significa que dominam todas as outras. A pesquisa de Christakis e Fowler mostrou que a disseminação da influência nas redes sociais obedece àquilo que chamaram de *Regra dos Três Graus de Influência* [30]. Tudo o que um indivíduo faz, tende a reverberar pela rede, exercendo um impacto que se dissipa a cada grau de separação, até um provável limite.

Eles concluíram que, apesar da possibilidade de conexão, além da fronteira dos três graus de separação, a influência humana para de ter efeito perceptível. Quando o objeto de estudo, portanto, é o contágio ou a influência, os três graus de separação parecem limitá-los em profundidade.

De acordo com a pesquisa, há três possíveis razões para essa influência ser limitada. Primeiro, existe deterioração na fidelidade das informações durante sua transmissão. Christakis e Fowler nominaram este motivo de *explicação do decaimento intrínseco* [30].

Segundo, a influência pode diminuir devido a uma transformação inevitável na rede que torna instáveis os elos além dos três graus de separação, dada a rotatividade constante nos laços sociais por todo o caminho. Para uma pessoa A que está a quatro graus de distância de uma pessoa B, qualquer um dos quatro laços intermediários pode ser cortado, e B perderá pelo menos um caminho até A. Consequentemente, para Christakis e Fowler, as pessoas não influenciam indivíduos a mais de três graus de separação. Este motivo foi nominado pelos pesquisadores de *explicação da instabilidade da rede* [30].

Por fim, a biologia evolutiva pode ser um fator preponderante. Os seres humanos talvez não sejam capazes de influenciar pessoas a quatro graus de distância, pois em seu passado hominídeo, não havia distâncias maiores do que três graus. Christakis e Fowler nominaram este motivo de *explicação do objetivo evolutivo* [30].

Assim, embora a observação de que há seis graus de separação entre duas pessoas quaisquer se aplique ao grau de conexão, a observação de que há três graus de influência se aplica ao grau de contágio. Para Christakis e Fowler, estas propriedades, conexão e contágio, são a estrutura e a função das redes sociais.

2.6 Modelagem de Redes

Não obstante sua sofisticação, simplificar todas as redes em grafos é algo que suscita alguns desafios formidáveis. Se a sociedade, a Internet, a célula ou o cérebro podem ser representados por grafos, cada um deles é claramente diferente entre si.

Como evidencia Barabási [14], é difícil imaginar que exista muito em comum entre a sociedade, na qual amigos e conhecidos são criados por intermédio de encontros casuais e decisões conscientes, e a célula, na qual leis inexoráveis da química e da física governam todas as reações entre as moléculas. Obviamente, deve haver uma diferença nas regras que governam a localização de *links* nas várias redes encontradas na natureza. Descobrir um modelo para descrever todos esses diferentes sistemas pode parecer, à primeira vista, um desafio intransponível, ousadamente aceito pela Ciência das Redes.

2.6.1 Universo Randômico

Na década de 50, os brilhantes matemáticos Paul Erdös e Alfréd Rényi aceitaram o desafio de descrever todos os grafos complexos com um único esquema, propondo uma formulação matemática. Como cada sistema, particularmente obedece a regras díspares

na configuração de sua rede, Erdös e Rényi deliberadamente propuseram a solução mais simples que a natureza poderia adotar: conectar os nós aleatoriamente [14].

Antes de Erdös e Rényi a Teoria dos Grafos não lidava com redes sociais ou grafos aleatórios, concentrando-se quase exclusivamente em grafos regulares, sem qualquer ambiguidade acerca de sua estrutura. Não obstante, quando o objeto de estudo são sistemas complexos, os grafos regulares são a exceção, e não a regra. Erdös e Rényi reconheceram que os grafos reais, das redes sociais às linhas telefônicas, não são regulares, e sim, irremediavelmente complicados [14]. Subjugados por sua complexidade, ambos os matemáticos admitiram que essas redes fossem randômicas.

Os grafos regulares são os únicos que garantem para cada nó, exatamente o mesmo número de *links*, formando verdadeiras redes igualitárias. Essa regularidade está claramente ausente nos grafos randômicos. Todavia, a premissa do modelo randômico é profundamente igualitária, pois como os *links* são estabelecidos de maneira aleatória, cada nó possui a mesma chance de obtê-los.

A Figura 2.8 [106] é um gráfico da fração da rede, ou do grafo aleatório, ocupado por seu maior componente conectado versus o número de *links* presentes. Watts mostrou, que quando se tem poucos *links*, nada se conecta [106]. Como as conexões são criadas de forma aleatória, quase sempre se conecta um nó isolado a outro e, mesmo que um deles já possua uma conexão, ela provavelmente só levará a um pequeno número de nós.

Intrigantemente, quando se acrescenta conexões para que cada nó tenha na média uma conexão, a fração do grafo ocupada pelo maior componente salta, rapidamente, de quase 0 para cerca de 1. Na física essa mudança repentina é conhecida como transição de fase, e o ponto em que isso começa a acontecer, é chamado de ponto crítico [106].

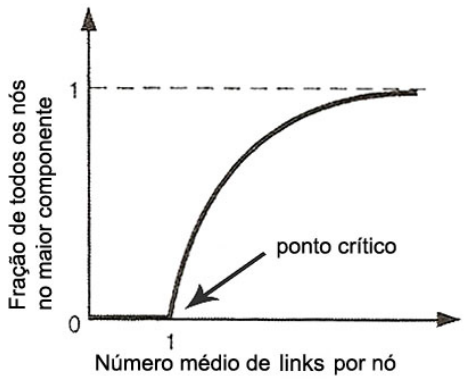


Figura 2.8: *Gráfico de Conexões Aleatórias.*

Dessa forma, Erdös e Rényi explicaram que 1 é o limiar. Se os nós tiverem menos de uma conexão em média, então a rede se fragmentará em pequenos aglomerados incomunicáveis, caso contrário esse risco tornar-se-á remoto.

Essa observação é relevante, pois se dois nós não são parte do mesmo componente, não podem se afetar mutuamente. Seria como se estivessem em sistemas diferen-

tes, com o comportamento de um não tendo nenhuma relação com o comportamento do outro [106]. A presença de um único componente, conhecido na literatura também por *componente gigante*, significa que tudo o que acontece em um ponto da rede pode potencialmente afetar qualquer outro ponto. Sua ausência, ao contrário, implica que eventos locais só serão sentidos localmente.

A natureza, via de regra, excede o mínimo de um *link*. Os sociólogos estimam que as pessoas conheçam entre 200 e 5 mil pessoas por nome. Um neurônio médio está conectado a dezenas de outros, alguns a milhares. No corpo humano, muitas moléculas participam de mais de uma única reação, algumas, como a água, de centenas.

Erdös descobriu que, independente da quantidade de nós, uma pequena porcentagem de ligações aleatórias é sempre suficiente para ligar a rede como um todo quase que completamente conectado. Ainda mais surpreendente, é que a porcentagem necessária diminui à medida que a rede cresce [23]. Em uma rede de 300 pontos, há cerca de 45.000 ligações possíveis entre eles, mas, não mais que 2% dessas ligações garantem à rede um componente gigante completamente conectado. Para 1.000 pontos, a fração crucial é de menos de 1%. Para 10 milhões de pontos, é de apenas 0,00016%.

Matematicamente, o resultado é que em uma rede com N vértices, a porcentagem necessária de ligações para amarrar a rede em um componente gigante é dada por:

$ln(N)/N$, onde *ln(N)* é o logaritmo natural de N [23].

Para uma rede de sete bilhões de pessoas, a fração revelada pelos cálculos de Erdös não seria mais do que 0,000000004, ou cerca de quatro em um bilhão. Este número implica que, se as pessoas realmente estivessem ligadas aleatoriamente, uma pessoa típica teria de conhecer cerca de uma pessoa a cada 305 milhões para que toda população do mundo estivesse ligada em uma teia social inteiramente conectada. No total, isso significa apenas 23 conhecidos para cada uma das pessoas, um número bastante baixo se comparado ao estabelecido pela sociedade atual. Sob esse raciocínio, não seria surpreendente que duas pessoas quaisquer no mundo pudessem estar conectadas através de um caminho de ligações sociais.

Dentro dessa perspectiva, no início de 1999, Barabási juntamente com Réka Albert e Hawoong Jeong propuseram uma fórmula matemática que prediz a separação em uma rede randômica como uma função do número de nós [16]. A origem da pequena distância revelada neste tipo de rede é devida à função logarítmica presente na fórmula. Para N nós em uma rede, que tenham em média k *links*, d *links* distantes, tem-se:

$$d = logN/logk \text{ [16]}$$

Dessa forma, em duas redes, ambas com uma média de 10 *links* por nós, sendo uma delas cem vezes maior do que a outra, a separação na rede maior será apenas dois graus superior à separação na rede menor.

Porém, por mais sofisticada que seja a teoria dos grafos aleatórios, as descobertas recentes sobre as redes reais, de redes sociais a redes neuronais, sugerem que elas não são nem um pouco similares aos grafos de Erdös e Rényi. Caso as pessoas realmente escolhessem seus amigos aleatoriamente, seria muito mais provável que elas fizessem amizade com alguém de outro continente do que com alguém em suas próprias vizinhanças.

Além disso, mesmo que as pessoas tivessem mil amigos, a chance de quaisquer dois amigos dessas pessoas se conhecerem seria grosseiramente uma em sete bilhões [106]. Pela experiência cotidiana, sabe-se que os amigos de uma pessoa tendem a se conhecer, logo, grafos aleatórios não podem ser uma boa representação do mundo social. Desta forma, para se entender as propriedades e o comportamento de redes no mundo real, a questão da estrutura não randômica teve de ser enfrentada.

2.6.2 Não Aleatoriedade

Contemporânea à pesquisa de Erdös e Rényi, o matemático Anatol Rapoport e seus colaboradores iniciaram uma pesquisa sobre redes aleatoriamente conectadas e tentaram enfrentar o que viram como falhas essenciais [86]. Incumbidos de compreender a disseminação de doenças na sociedade, os pesquisadores logo perceberam que para algumas delas, a rede subjacente é crítica.

Dentre algumas indagações que o grupo de Rapoport levantou, destacam-se:

- Algumas relações deveriam ser reconhecidas como mais importantes do que outras?
- A maioria das pessoas tem aproximadamente o mesmo número de amigos, ou algumas têm muito mais amigos do que a média?
- Como se explica a existência de grupos, dentro dos quais os laços de amizade são densos, mas entre os quais as conexões são relativamente esparsas?

Tais indagações provavam que a não aleatoriedade era muito mais difícil de definir, já que praticamente todos os grafos aleatórios são essencialmente os mesmos.

A homofilia, tendência consciente ou inconsciente de se associar a pessoas similares, de certa forma ajuda a explicar como as pessoas se conhecem em sociedade. Não obstante, a pesquisa de Rapoport se concentrava na questão de como os indivíduos que as pessoas conhecem hoje determinam os que elas conhecerão no futuro. Para isso, ele introduziu o conceito de *pensamento triádico* [86].

Em redes sociais, o nível seguinte em simplicidade da díade, e base de toda estrutura de grupo, é um triângulo, ou tríade, que surge sempre que um indivíduo tem dois amigos que também se conhecem. A inclusão da dinâmica nesta perspectiva conduz à situação natural, de que dois estranhos que possuam um amigo em comum tenderão a

se conhecer com o tempo [86]. Redes sociais, portanto, ao contrário de redes aleatórias, evoluirão de forma que as tríades tendem a se fechar sobre si mesmas.

Rapoport identificou que a aleatoriedade falha em capturar alguns dos princípios ordenadores mais poderosos que também governam as escolhas que as pessoas fazem. Percebendo que deveria de alguma forma, equilibrar esses dois conjuntos de forças em um único modelo, o matemático concluiu que as redes reais seguem alguns princípios ordenadores importantes, sendo em todo o resto, aleatórias. Ele chamou sua nova classe de modelos de *redes aleatórias inflexionadas* [106].

Nas redes sociais, as pessoas agem em parte devido à posição estrutural que ocupam, e em parte devido às suas preferências e características inatas. A estrutura da rede ao tecer uma teia social, faz com que os indivíduos que uma pessoa conhecerá amanhã dependam, ao menos em algum nível, dos que ela já conhece hoje. Todavia, as características intrínsecas dos indivíduos os levam a conhecer pessoas que não têm conexão alguma com seus amigos anteriores.

Portanto, a evolução de uma rede social é impelida por um jogo entre essas duas forças, conhecidas em sociologia por *estrutura* e *agência*. Como a agência é a parte do processo de decisão de um indivíduo restrita às suas características intrínsecas, ações derivadas da mesma parecem eventos aleatórios [106].

Uma vez que as afiliações aparentemente aleatórias tenham sido feitas, a estrutura novamente entra em cena, e as intersecções recém-criadas se tornam as pontes por onde outros indivíduos podem atravessar, formando novas afiliações.

2.6.3 Pontes entre Mundos

Ao contrário da maioria dos pesquisadores de redes sociais, o sociólogo Mark Granovetter investigou qual poderia ser o segredo dos mundos pequenos, inspirado nos trabalhos de Rapoport [50]. Os trabalhos sobre redes sociais anteriores ao de Granovetter, não diferenciaram a força das amizades ou dos laços entre as pessoas, que obviamente possuem diferenças de intensidade. Em termos gerais, o sociólogo denominou de laços fortes, aqueles que existem entre membros de uma mesma família, ou entre bons amigos, ao passo que laços fracos unem apenas conhecidos.

Vários autores que tentaram operacionalizar as diferenças entre os dois tipos de laços, verificaram a existência dos seguintes traços característicos [66]:

- Os laços fortes dão origem a relações mais frequentes do que os laços fracos, sendo os serviços recíprocos mais comuns nos primeiros;
- Há igualmente mais intensidade emocional nos laços fortes do que nos laços fracos;
- A multiplexidade da relação é maior nos laços fortes, o que significa que os amigos então ligados entre si em áreas mais diversas do que os conhecidos.

A grande introspecção de Granovetter foi perceber que o fechamento triádico, percebido por Rapoport, só é concretizado quando as pessoas estão fortemente ligadas. Em um grafo, esse princípio implica que os laços fortes não existem isoladamente. Desta forma, eles tendem a aparecer em triângulos, como exemplificado na Figura 2.9, e triângulos incompletos devem ser raros [23].

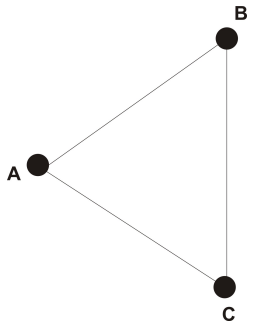


Figura 2.9: *Triângulo Social.*

O cientista social Ronald Burt denominou esses triângulos incompletos de *buracos estruturais*, afirmando que os mesmos são encontrados essencialmente nos grupos onde predominam os laços fracos [24]. Consequentemente, os buracos estruturais são numerosos nos grupos de fraca densidade em que se verifica a ausência de conexões diretas entre vários pares de indivíduos. A Figura 2.10, ilustra a situação onde o nó *A* se encontra na posição de buraco estrutural.

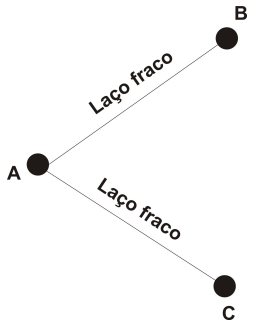


Figura 2.10: *Buraco Estrutural.*

Essa ideia, aparentemente irrelevante, conduz a uma conclusão paradoxal. Se um laço forte de uma rede social for removido, isso causará pouco efeito no número de graus de separação. Como os laços fortes quase sempre surgem em triângulos, ainda seria possível ir de uma ponta a outra do elo rompido, em apenas dois passos, movendo-se pelas duas bordas restantes do triângulo.

O que torna as coisas paradoxais é que, por intuição, laços fortes parecem ser os laços responsáveis por manter a rede coesa. Mas quando se trata dos graus de separação de uma rede, os laços cruciais, como Granovetter veio a demonstrar, são os laços fracos entre as pessoas, especialmente os que ele chamou de pontes sociais [50].

No contexto social, uma ponte têm o significado de um caminho que proporciona fácil acesso entre dois indivíduos. Na ilustração de Buchanan, verificada na Figura 2.11 [23], laços fortes são representados por linhas contínuas, e laços fracos por linhas pontilhadas. O número de graus de separação entre os indivíduos *A* e *B* salta de um para oito ao se remover o único laço entre eles, demonstrando a importância das pontes sociais.

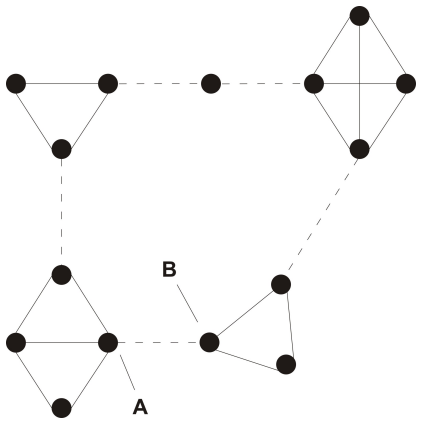


Figura 2.11: *Pontes Sociais.*

Dessa forma, Granovetter chegou a uma conclusão surpreendente: laços fracos têm frequentemente mais importância do que laços fortes, porque agem como ligações cruciais que costuram o tecido social. São eles os responsáveis por criar os atalhos que ao serem eliminados, levam a rede a se desmanchar em componentes desconectados.

Para Granovetter, pontes entre mundos propiciam um arranjo estrutural, que certamente favorece a condição de existência de um mundo pequeno. Se os seis graus de separação realmente são verdade, atalhos sociais devem cumprir algum papel crucial.

O argumento de Granovetter propõe uma imagem de sociedade bem diferente do universo randômico de Erdös e Rényi. Se sua esquematização estiver correta, trata-se se uma coleção de grafos completos, unidos uns aos outros por algumas ligações fracas entre conhecidos que pertencem a diferentes círculos de amigos. A ilustração de Barabási apresentada na Figura 2.12 [14], demonstra claramente este tipo de arranjo social.

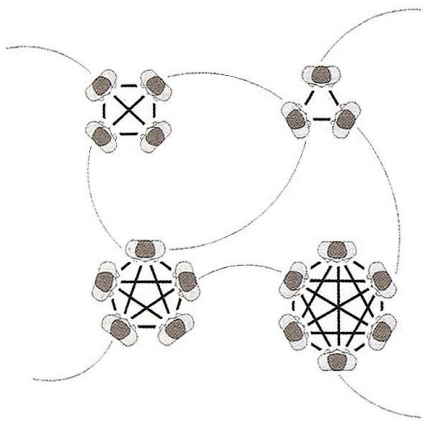


Figura 2.12: *A Sociedade de Granovetter.*

Para entender plenamente a estrutura da sociedade, a teoria das redes randômicas deveria ser conciliada, de certa forma, com a realidade das aglomerações descritas por Granovetter. Combinada à descoberta de Milgram dos seis graus de separação, estas linhas de pensamento preparavam o terreno para uma revolução que hoje está repercutindo em áreas tão díspares entre si quanto a medicina, a sociologia e a economia.

2.6.4 O Modelo Alfa

Inconformados com a resignação ao problema do mundo pequeno por parte da comunidade científica, os pesquisadores da Universidade de Cornell, Duncan Watts e Steven Strogatz, olharam para o problema sob uma nova perspectiva. Ao invés de se preocuparem sobre quão pequeno era o mundo, eles pesquisaram o que era necessário para que um mundo se tornasse pequeno. Consequentemente, eles se propuseram a construir um modelo de rede social, aplicando sobre ele a força da matemática e dos computadores.

Em vez de tentar determinar o equilíbrio ideal entre desejo individual e estrutura social pré-identificados por Rapoport no mundo real, Watts e Strogatz começaram a examinar diversas perspectivas possíveis. Sob este novo prisma, eles definiram a importância relativa da ordem e da aleatoriedade como parâmetros a serem ajustados, a fim de percorrer um espaço de possibilidades.

Por conseguinte, os pesquisadores revelaram uma maneira engenhosa de conectar as ideias de Granovetter às de Milgram. Eles descobriram uma forma sutil de ligar os pontos no grafo que não era nem ordenada nem aleatória, mas algo entre estas duas formas, um padrão incomum no qual o caos e a ordem se misturavam em equilíbrio [23]. Analisando variações deste grafo intermediário, Watts e Strogatz descobriram que ele era a chave para revelar o enigma dos seis graus de separação.

Para tornar compreensível o modelo, Watts e Strogtaz propuseram a criação de dois mundos distintos [106]. No primeiro, nomeado por eles de *cavernas*, as relações são vividas em laços coesos auto-reforçados, deixando praticamente inconcebível o início de uma relação entre indivíduos estranhos. No segundo, intitulado *Solaria*, as interações são igualmente acessíveis, fazendo com que relacionamentos anteriores sejam relativamente irrelevantes para o estabelecimento de novos laços sociais.

Para criarem esses mundos, e alicerçados sobre o modelo de sociedade previsto por Granovetter, os pesquisadores pensaram sobre a probabilidade de uma pessoa conhecer outra, em função de quantos amigos possuam em comum em um dado momento. Para tanto, introduziram uma grandeza chamada *coeficiente de aglomeração* [106].

Para exemplificar o cálculo dessa métrica, Watts ilustra a situação em que alguém tem quatro amigos, que também são amigos entre si. Desta forma, obtêm-se ao todo seis *links* de amizade. Se alguns desses quatro amigos não forem amigos uns dos outros, a

conta chegará a menos de seis *links*. Caso a quantidade de *links* estabelecida fosse quatro, o coeficiente de aglomeração do círculo de amigos dessa pessoa seria 0,66, que se obtém dividindo-se o número de *links* entre os amigos de alguém, pelo número de *links* possíveis de amizade [106].

O coeficiente de aglomeração informa o nível de coesão do círculo de amigos dos indivíduos. Pela ótica de Granovetter, a sociedade deve comportar inúmeras aglomerações, o que consequentemente produz um alto valor para esta métrica.

Seguindo esse raciocínio, Watts definiu que no mundo das *cavernas*, uma ausência de conhecidos em comum, sugere vidas em cavernas diferentes, logo, provavelmente as pessoas nunca se encontrarão. Mas, se elas tiverem pelo menos um amigo em comum, a implicação é que vivem na mesma comunidade, possuindo extrema probabilidade de se conhecer. No outro extremo, em *Solaria*, a história social é irrelevante para o futuro das pessoas. Mesmo que duas pessoas possuam vários amigos em comum, terão quase a mesma probabilidade de se conhecer caso possuam nenhum [106].

Os dois tipos de mundo podem ser capturados pelas regras apresentadas no modelo da Figura 2.13 [106]. Neste modelo, verifica-se que a tendência de duas pessoas se tornarem amigas é determinada pelo número atual de amigos em comum, mas que a maneira pela qual esta tendência é determinada difere drasticamente entre as duas regras.

Esse foi o primeiro modelo matemático de uma rede social, batizado por Watts de *modelo alfa* e, assim, seu parâmetro ajustável de controle se tornou *alfa*. Entre os dois extremos, uma família de regras de interação, especificada por um valor particular para α. Quando $\alpha = 0$, tem-se o mundo das *cavernas*, quando α se torna infinito, *Solaria*[106].

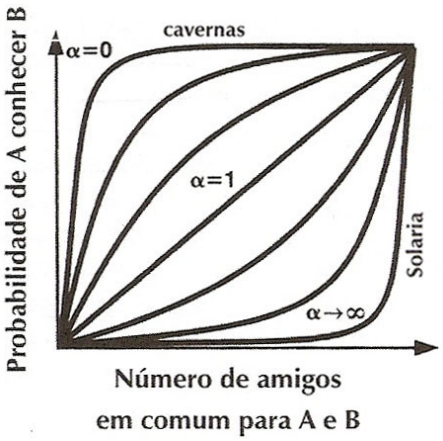


Figura 2.13: *Modelo Alfa.*

A origem do fenômeno do mundo pequeno parecia depender da presença de dois mundos contraditórios [106]. O mundo das *cavernas* era bastante aglomerado, o que dificultava conectar todas as pessoas do mundo em apenas alguns passos. Em contraste, o mundo de *Solaria* tinha muito mais probabilidade de exibir caminhos curtos na rede,

pois um resultado padrão da teoria dos grafos aleatórios é que, em média, o comprimento típico dos caminhos é pequeno.

No entanto, em um grafo aleatório, a probabilidade de quaisquer dois amigos de uma pessoa se conhecerem se torna desprezível para uma população muito grande, o que consequentemente, deixa o coeficiente de aglomeração com valores baixos. Este contexto, portanto, sugeria que um mundo poderia ser pequeno ou aglomerado, mas não as duas coisas simultaneamente, evidenciando o intrigante paradoxo das redes sociais.

2.6.5 Redes de Mundo Pequeno

Através da Figura 2.14 [106], Watts ilustrou o paradoxo das redes sociais. O comprimento típico dos caminhos é pequeno quando o valor de *alfa* é baixo, se computado apenas entre nós de um mesmo componente. Similarmente, este comprimento apresenta o mesmo comportamento para valores altos de *alfa*. Todavia, para valores intermediários do parâmetro, o valor do comprimento dos caminhos dispara indiscriminadamente.

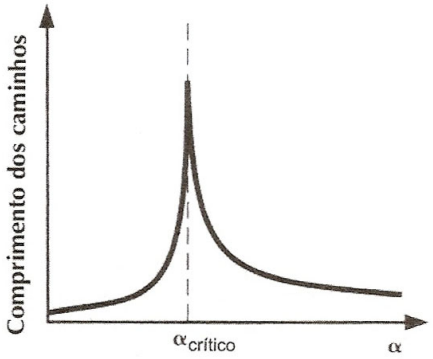


Figura 2.14: *Comprimento dos Caminhos no Modelo Alfa.*

À esquerda do pico da Figura 2.14, as diversas *cavernas* estão se juntando, resultando em um aumento dos comprimentos de caminho. O mundo torna-se maior, pois os componentes anteriormente isolados começaram a se conectar. Naturalmente, cada vez mais pessoas podem ser contatadas, tornando este processo um pouco mais complicado.

A direita do pico da Figura 2.14 representa a região onde todos os componentes da rede se conectaram em um componente gigante. O comprimento dos caminhos decai à medida que a regra de interação se torna mais aleatória. O pico em si representa o ponto crítico, no qual todo mundo fica conectado, mas os comprimentos típicos dos caminhos tendem a ser grandes.

O grande *insight* de Watts e Strogatz ocorreu quando os pesquisadores verificaram a localização da transição do coeficiente de aglomeração relativamente à transição correspondente do comprimento dos caminhos. Eles perceberam que estes pontos de transição não eram correspondentes, como mostra a Figura 2.15 [106].

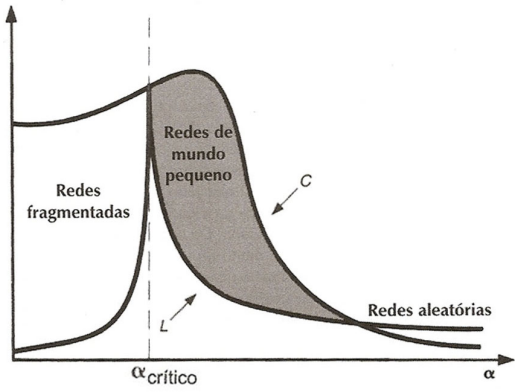


Figura 2.15: *Coeficiente de Aglomeração.*

Como os pesquisadores concluíram, a compreensão dessa perspectiva é que o coeficiente de aglomeração, *C*, não cai na mesma taxa do comprimento dos caminhos, *L*. Sob esta conjectura, as redes reais seriam capazes de exibir alta aglomeração, além de uma pequena distância entre seus componentes. Watts e Strogtaz denominaram essa classe de redes de *redes de mundo pequeno*.

A comparação entre comprimentos de caminho e coeficiente de aglomeração, realizada na Figura 2.15, mostra que a região sombreada entre as curvas, onde *L* é baixo e *C* é alto, representa a presença de redes de mundo pequeno. O parâmetro *alfa* representa um equilíbrio entre as restrições da estrutura social e a liberdade da agência individual. Como no mundo real, cada pessoa pode se conectar a qualquer outra, na visão dos pesquisadores, este mundo certamente fica à direita do pico da Figura 2.15.

Mas por não saberem exatamente o que um valor específico de *alfa* poderia significar, Watts e Strogatz perceberam que era necessário, encontrar uma forma simplificada de ajustar cada rede entre a completa ordem e a completa desordem. Por conseguinte, começaram a interpolar redes de modo que se passasse por todos os estágios intermediários em um novo modelo, designado *modelo beta*.

Criaram inicialmente, um reticulado regular como o mostrado à esquerda da Figura 2.16 [108], no qual cada nó está conectado a um número fixo de vizinhos mais próximos no anel. Em seguida, escolheram elos aleatoriamente e começaram a religá-los.

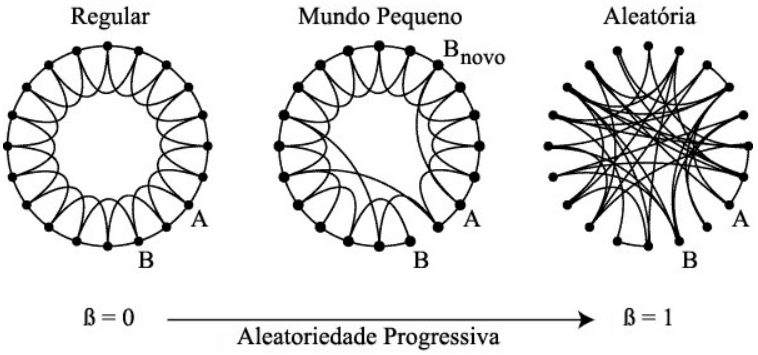


Figura 2.16: *Modelo Beta.*

Na prática, o que eles fizeram foi dar um valor, entre zero e um, ao novo parâmetro ajustável *beta* e, então, visitaram cada elo no reticulado, religando-os aleatoriamente, com probabilidade *beta* [108]. Assim, quando $\beta = 0$, nenhuma religação acontece, e as pessoas acabavam ligadas como no reticulado regular inicial. No outro extremo, quando $\beta = 1$, todos os elos são religados, e o resultado é uma rede muito próxima a um grafo aleatório, completamente desordenada.

A grande indefinição era o que acontecia no meio desses extremos. Com uma pequena probabilidade de religação, ilustrada no centro da Figura 2.16, o objeto resultante assemelha-se a um reticulado regular, se diferenciando pela presença de algumas conexões aleatórias. Quando Watts e Strogatz examinaram o coeficiente de clusterização, alguns poucos elos aleatórios não faziam muita diferença, pois a maioria dos amigos das pessoas ainda se conhecia, logo, a aglomeração continuava alta.

Não obstante, o comprimento dos caminhos diminuía dramaticamente, pois como os elos eram ligados uniformemente de forma aleatória, e como, em um grande reticulado, havia muito mais pontos longe dessa pessoa do que perto, a probabilidade era que ela fosse conectada a alguém que estava distante [108]. E, ao fazê-lo, não apenas estes indivíduos ficavam mais perto entre si, como também grandes pedaços do resto da rede ficavam muito mais próximos.

A principal descoberta dos pesquisadores consiste na verificação de que apenas alguns elos aleatórios, o que implica em uma pequena possibilidade de agência individual, podem gerar um efeito surpreendente, visualizado na Figura 2.17 [106]. Quando *beta* deixa de ser zero, o comprimento dos caminhos, *L*, entra em queda livre, despencando exponencialmente, quase que como um eixo vertical. Consequentemente, ao diminuir a distância entre os nós, cada atalho reduz o efeito marginal dos atalhos subsequentes [106].

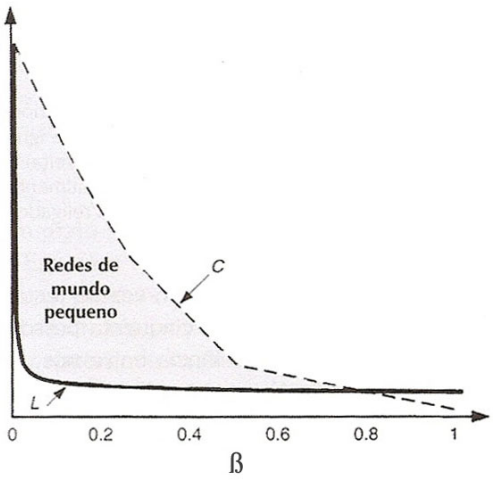


Figura 2.17: *Impacto das Religações Aleatórias no Modelo Beta.*

Nesse modelo simples, outro resultado surpreendente observado pelos pesquisadores é que, na média, as primeiras cinco religações aleatórias reduzem o comprimento

médio dos caminhos da rede pela metade, independentemente do seu tamanho [106].

Assim como Erdös descobriu que, independente da quantidade de nós, a porcentagem de ligações aleatórias para ligar a rede em um grafo conexo diminui à medida que a rede cresce, Watts e Strogatz identificaram que quanto maior a rede, maior o efeito da adição de cada elo aleatório para a diminuição do comprimento dos caminhos. Desta forma, o impacto do acréscimo de elos se torna, na prática, independente do tamanho.

Essas redes intermediárias estavam realmente conseguindo unir o melhor de dois mundos, ser um mundo pequeno e altamente aglomerado. Uma rede com dez pessoas, ligadas de maneira puramente aleatória, possui as propriedades de redes de mundo pequeno. Mas, para 1.000 pessoas ligadas aleatoriamente, o coeficiente de aglomeração é de cerca de 0,006, o que não chega nem perto do que se encontra em uma rede social real.

O que Watts e Strogatz provaram com o *modelo beta*, foi que a consequência inevitável da incorporação de ordem e algum nível de desordem por parte dos sistemas conectados, será a formação das redes de mundo pequeno. Como pista de uma aparente universalidade deste processo, os pesquisadores perceberam que este tipo de rede deveria surgir não apenas no mundo social, mas em outros tipos de rede encontrados na natureza.

Não muito tempo depois que os estudantes de computação Glen Wasson e Brett Tjaden disponibilizaram o *site The Oracle of Kevin Bacon* [105], Watts e Strogatz tiveram acesso à base de dados utilizada no projeto. Este *site* é capaz de calcular a distância entre dois atores quaisquer de *Hollywood*, listando a cadeia de filmes e atores pelos quais estão conectados. De posse desta base de dados, os pesquisadores puderam calcular o comprimento médio dos caminhos e o coeficiente de aglomeração de toda a rede.

O resultado pode ser visualizado na Tabela 2.1 [108]. Em um mundo com milhares de indivíduos, cada ator pode ser conectado a qualquer outro em uma média de menos de quatro passos. Além disso, os coadjuvantes de cada ator tinham alta probabilidade, 80% das vezes, de haver estrelado juntos em algum filme. Para Watts e Strogatz não havia dúvida de que aquela era uma rede de mundo pequeno [108].

Tabela 2.1: *Mundos Pequenos*

	L_{real}	$L_{aleatório}$	C_{real}	$C_{aleatório}$
Atores de Cinema	3,65	2,99	0,79	0,00027
Rede Elétrica	18,70	12, 40	0,080	0,005
C. elegans	2,65	2,25	0,28	0,05

L=Comprimento dos Caminhos e C=Coeficiente de Aglomeração.

Curiosos para entender como as redes sociais se distinguem de outros tipos de rede, Watts e Strogatz começaram a estudar a rede de energia elétrica dos Estados Unidos e a rede de neurônios do nematódeo *Caenorhabditis elegans*, uma criatura tão simples que os biólogos dos anos 80 foram capazes de mapear seu sistema nervoso inteiro [108].

A rede elétrica americana foi criada pela mente humana, e o sistema nervoso de um nematódeo, pela evolução. Não obstante, ambas as redes provaram ter quase exatamente a mesma estrutura apresentada pelo universo social, como também pode ser visualizado na Tabela 2.1. Como afirma Buchanan, "por alguma razão, os grafos de Watts e Strogatz pareciam apontar para algum princípio organizador do mundo" [23].

Assim, Watts e Strogatz tinham agora três exemplos de rede que, não apenas satisfizeram a condição de mundo pequeno, como o fizeram apesar de suas enormes diferenças de tamanho, densidade e, mais importante, natureza básica.

Subsequentemente, a comunidade científica começou a estudar vários outros tipos distintos de rede. As propriedades de redes de mundo pequeno já foram identificadas na Web e nas redes físicas que conectam os computadores na Internet. Os biólogos as descobriram na rede de moléculas envolvidas em uma célula. Os ecologistas as encontraram nas cadeias alimentares. Os cientistas da computação as observaram nas redes de colaboração entre cientistas. A descoberta da aglomeração alçou-a de característica específica da sociedade à posição de propriedade genérica das redes complexas, subvertendo a tratativa de que estas redes são randômicas.

2.6.6 Redes Sem Escala

Como as motivações de Watts e Strogatz vinham de um estudo sobre mundos pequenos, eles não se interessaram pela análise da quantidade de conexões que os indivíduos possuem em suas redes [106], subjugando esta informação como irrelevante.

A distribuição de graus de uma rede apresenta em uma única imagem, como ilustrado na Figura 2.18, a probabilidade $p(k)$ de um membro da rede, aleatoriamente selecionado, ter um número específico k de amigos, ou seu relativo grau de conexão.

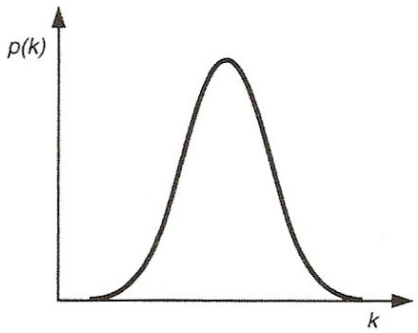


Figura 2.18: *Distribuição Normal.*

A suposição que Watts e Strogatz fizeram sobre as redes de forma geral, foi que elas teriam uma distribuição de graus semelhante à da Figura 2.18, conhecida como curva de sino [106]. Ou seja, não apenas existiria um grau médio definido, como a maioria dos nós também teria graus não muito distantes da média.

A distribuição deste tipo de curva decai de forma extremamente rápida em ambos os lados do pico pronunciado, geralmente em escala exponencial. Desta forma, a probabilidade de qualquer indivíduo ter mais amigos do que a média torna-se desprezível.

Esse tipo de distribuição satisfazia uma importante exigência dos modelos de Watts e Strogatz, que ninguém na rede estivesse conectado a nada além de uma pequena fração da população inteira [106]. Diversas distribuições no mundo real, inclusive, as concernentes aos grafos aleatórios, possuem exatamente essas propriedades, motivo pelo qual usualmente esta distribuição é conhecida como *distribuição normal*.

Quando o trabalho pioneiro de Watts e Strogatz foi publicado, o grupo de pesquisa de Barabási estava tentando entender a estrutura das redes complexas, enfocando a Web. Para isso, os cientistas construíram um computador "robô", *crawler*, programado para surfar pela rede. O *crawler* começava por um *site* qualquer, reunindo todos os *links* existentes no mesmo. Em seguida, ele seguia cada um dos *links* isoladamente, e repetia o procedimento para cada nova página encontrada.

Ao encaixarem o histograma da conectividade de um nó em um gráfico *log-log*, Barabási e seus colaboradores verificaram que a distribuição de *links* em várias páginas da Web seguia uma expressão matemática denominada *lei de potência* [15].

Leis de potência são outro tipo de distribuição dos sistemas naturais, com dois diferenciais cruciais perante as distribuições normais. Primeiro, uma lei de potência não tem um pico em seu valor médio, começando no valor máximo e decrescendo até o infinito, como exemplificado na Figura 2.19. Segundo, o ritmo em que a lei de potência decai é muito mais lento do que o da distribuição normal, constituindo sua "cauda longa".

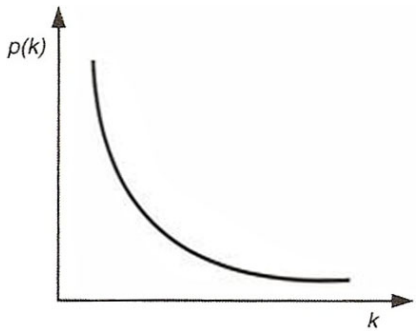


Figura 2.19: *Distribuição por Lei de Potência.*

Comparadas com as curvas de sino, as curvas baseadas em leis de potência tendem a zero mais lentamente, o que implica uma maior probabilidade de existirem nós dotados de um número realmente grande de *links*. O grupo de Barabási nomeou estes nós bem conectados de *hubs*, constatando que os mesmos detêm 80 a 90% do número total de ligações da rede, em detrimento de uma maioria de nós com poucas conexões [15].

Após a descoberta sobre a Web, ao analisarem a rede de atores de *Hollywood*, o grupo de Barabási verificou que o número de atores que tinham *links* com exatamente

k outros atores decaía seguindo uma lei de potência. A rede existente no interior da célula integrou a lista quando verificaram que o número de moléculas que interagem com exatamente k outras moléculas decresce seguindo uma lei de potência. O mesmo ocorreu com a distribuição de graus da rede de citações científicas [15].

O *expoente* de uma distribuição por lei de potência rege como a mesma muda em função da variável subjacente. Se o número de elementos com específico grau de conexão decai na razão inversa do seu grau de conexão, a distribuição tem expoente 1. Caso a distribuição decresça na razão inversa do quadrado do tamanho, o expoente será igual a 2.

Subsequentemente, em diversas redes que o grupo de Barabási e outros cientistas tiveram a oportunidade de investigar, surgiu um padrão incrivelmente simples. O número de nós com exatamente *k links* obedece a uma lei de potência, cada qual com um expoente que, para a maioria dos sistemas, varia entre 2 e 3. Isso implica que o número de páginas da Web com exatamente *k links* apontando para si, denotado por *N(k)*, é dado por [5]:

$$N(k) \approx k^{-y}, \text{ sendo o } \textit{expoente } y \text{ para a Web} \approx 2{,}1.$$

Percebe-se melhor as diferenças visuais e estruturais entre redes randômicas e regidas por leis de potência, ao se comparar os mapas rodoviários e de linhas aéreas dos Estados Unidos, como precisamente ilustrou Barabási na Figura 2.20 [14].

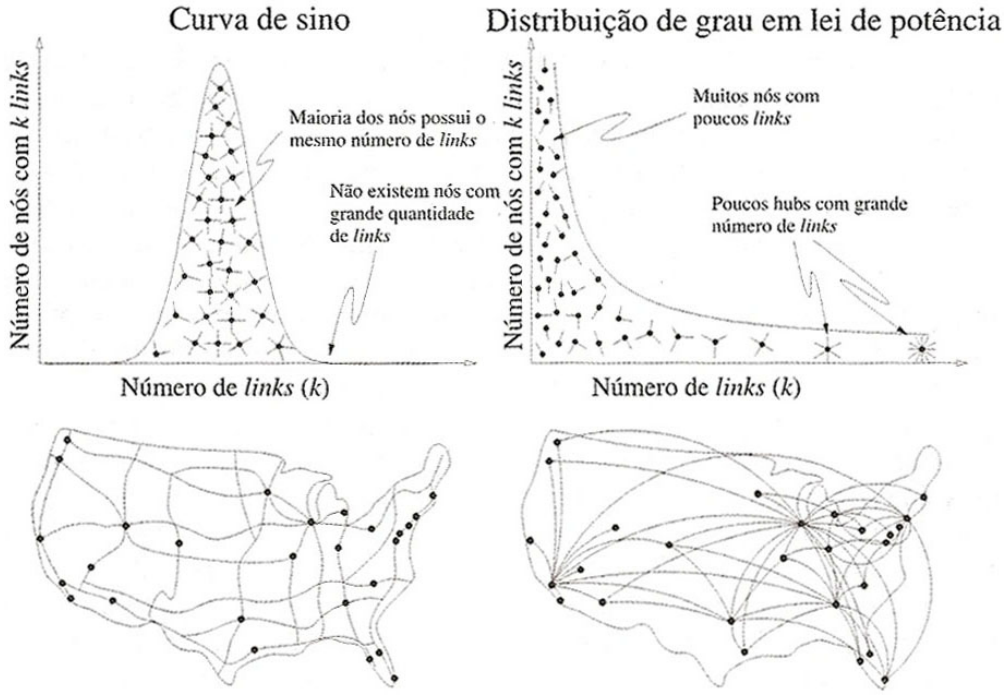


Figura 2.20: *Mapas Rodoviários e de Linhas Aéreas dos EUA.*

Nos mapas rodoviários, as cidades são os nós e as estradas que as conectam, os *links*. Esta rede apresenta características de distribuição uniforme, pois cada cidade grande tem pelo menos um *link* com o sistema rodoviário, e não há cidades servidas por

centenas de estradas [14]. Desta forma, muitos nós possuem aproximadamente o mesmo número de *links*, propriedade inerente às redes randômicas.

O mapa das linhas aéreas difere fortemente do mapa rodoviário. Na rede aérea, os nós são aeroportos conectados por voos diretos. A maioria dos aeroportos é pequena, representando nós que possuem poucos *links*, que por sua vez, os conectam com um ou diversos *hubs*, dos quais partem voos para quase todos os outros aeroportos do país [14].

A ausência de um pico em uma distribuição de grau por lei de potência implica a inexistência de um nó característico neste tipo de rede, que apresenta uma contínua hierarquia de nós, desde os raros *hubs* aos numerosos nós menores.

A atenção para os *hubs* é merecida, pois eles dominam a estrutura de todas as redes nas quais estão presentes [14]. De fato, com *links* para uma quantidade extraordinariamente grande de nós, os *hubs* criam atalhos entre dois nós quaisquer do sistema, definindo sob sua perspectiva, verdadeiros mundos pequenos.

Subjugada inicialmente por sua irrelevância, a distribuição de grau das redes reais passou a partir de então, a ocupar o seu devido lugar nos estudos sobre redes e sistemas complexos. Diante das comprovações expostas pelo grupo de pesquisa de Barabási, as leis de potência obrigavam os cientistas a abandonar a ideia de uma escala ou de um nó característico nas redes estudadas. Por esse motivo, elas começaram a ser descritas pelo próprio grupo como *redes sem escala*, ou *redes livres de escala* [15].

A descoberta das leis de potência por si só já era muito importante, mas o que realmente atraiu o interesse da comunidade científica, foi o mecanismo de crescimento, proposto por Barabási juntamente com Réka Albert, pelo qual as redes poderiam evoluir no tempo [106]. O modelo de Watts e Strogatz propôs uma explicação simples para a aglomeração, colocando-a junto com as redes randômicas sobre a mesma égide. Não obstante, este modelo não esclareceu como surgem os *hubs*.

Inicialmente, Barabási percebeu que a maioria das redes reais compartilha um traço essencial: o *crescimento* [15]. Qualquer tipo de rede, certamente começa com poucos nós, cresce de forma incremental adicionando novos nós, e atinge o tamanho atual.

Tanto o modelo de Erdös e Rényi quanto o de Watts e Strogatz não previam esse crescimento, pois se preocupavam somente com as ligações entre um número fixo de nós. Ao tentar explicar a existência dos *hubs*, Barabási e seus colaboradores incorporaram o crescimento aos modelos de rede [106]. No entanto, eles demonstraram que somente esta propriedade não explica a emergência das leis de potência.

A conclusão do grupo do cientista foi perceber que nas redes reais, a conexão nunca é aleatória, pois ela depende da popularidade. "As páginas da Web com mais conexões têm maior probabilidade de serem conectadas novamente, pessoas mais conhecidas fazem mais novos amigos" [14]. Por esta perspectiva, a evolução das redes é governada pela *conexão preferencial* [15]. Guiados por ela, inconscientemente os indivíduos adicio-

nam *links* a uma taxa mais elevada para escolher nós que já são altamente conectados.

Encaixando as peças do quebra-cabeça, Barabási e Albert descobriram que as redes são governadas por duas leis: *crescimento* e *conexão preferencial*. Para responder se estas leis são suficientes para explicar os *hubs* e as leis de potência, ambos os pesquisadores propuseram um modelo de rede que incorporava as duas leis, através de um algoritmo definido pelas regras seguintes [15]:

- **Crescimento:** para cada iteração, adiciona-se um novo nó à rede;
- **Conexão preferencial:** cada novo nó deve se conectar aos nós existentes com dois *links*, sendo a probabilidade de escolha proporcional ao número de *links* dos nós.

Esse modelo representado na Figura 2.21, combinando crescimento e conexão preferencial, foi a primeira tentativa bem sucedida de explicar os *hubs*. As simulações computadorizadas de Albert logo indicavam que ele sempre gerava leis de potência, sendo posteriormente denominado pelo grupo de pesquisa como *modelo sem escala* [14].

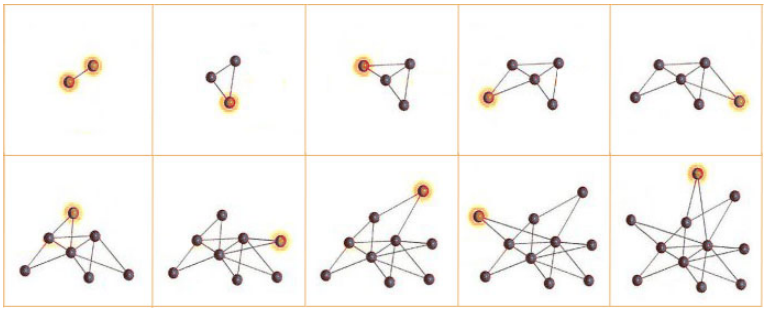


Figura 2.21: *Modelo Sem Escala.*

Albert rodou esse processo inúmeras vezes, e mesmo mudando o número de nós iniciais ou de *links* adicionados a cada iteração, a distribuição de graus permaneceu a mesma. As redes que emergiam eram sempre similares em termos de arquitetura. Todas eram mundos pequenos, com caminhos curtos entre os elementos, apresentavam aglomeração, e revelavam a peculiaridade dos *hubs*, como ilustrado na Figura 2.22.

Figura 2.22: *Rede Gerada pelo Modelo Sem Escala.*

O crescimento desempenha papel importante, pois a expansão da rede significa que os primeiros nós dispõem de mais tempo do que os últimos para adquirir *links*.

A antiguidade, contudo, como concluiu Barabási, não explica sozinha as leis de potência. Os *hubs* requerem a ajuda da conexão preferencial. Como novos nós preferem conectar-se a nós mais conectados, os primeiros nós com mais *links* serão escolhidos com mais frequência. Assim, eles crescerão mais rapidamente do que os nós mais jovens e menos conectados, criando um fenômeno conhecido como *rico fica mais rico* [15].

Um processo semelhante parece estar por trás do crescimento das redes sociais. No experimento original de Milgram, as cartas que conseguiram chegar à pessoa-alvo não percorreram a última etapa até o destino final vindas de qualquer lugar. De todas as que chegaram, dois terços foram enviadas por um único contato [14]. Estas pessoas são os *hubs*, os eixos altamente conectados do mundo social.

A simples generalidade do modelo de Barabási e Albert prometia uma nova forma de entender a estrutura das redes como sistemas em evolução dinâmica, não importando sua natureza. Enquanto o sistema obedecesse aos dois princípios básicos de crescimento e conexão preferencial, a rede resultante seria livre de escala.

Se a descoberta de Watts e Strogatz foi um primeiro passo para a compreensão do mundo das redes complexas, o reconhecimento dos *hubs* e das leis de potência para a distribuição das ligações foi um segundo passo igualmente importante.

2.6.7 Redes por Afiliação

Da mesma forma que Barabási percebeu a falha de Watts e Strogatz em reconhecer os *hubs* e as leis de potência, Watts igualmente identificou falhas nos modelos livres de escala. A limitação essencial deste último modelo, é que os laços são tratados como processos sem custo, podendo um nó ter tantos *links* quanto é capaz de acumular [106]. Sendo assim, o modelo adota uma visão que desconsidera restrições de limitação relativa aos nós, assumindo que os mesmos não possuem dificuldades de criar e manter conexões.

Para Watts, apesar de ter ficado claro o efeito das conexões preferenciais em relação à Web, o mesmo processo conectivo não poderia ser efetivamente garantido para outros tipos de redes, como sistemas humanos, biológicos, ou de engenharia.

Além disso, em uma sociedade, uma vez que as pessoas tenham feito seus contatos iniciais, a estrutura social em que estão inseridas torna alguns indivíduos mais acessíveis que outros, independente de seus respectivos graus de conexão. Este era o efeito que Watts e Strogatz tentaram capturar em seus modelos de mundos pequenos, pois os modelos livres de escala não tinham elementos de estrutura e dinâmica social.

Por conseguinte, juntamente com Steven Strogatz, Mark Newman e Peter Dodds, Watts iniciou o projeto de um novo modelo de redes, mas que desta vez, não se baseasse

em reticulados artificiais [106]. Não obstante, assim que a métrica do reticulado era removida, não existia mais a possibilidade de se determinar a distância entre os nós e, portanto, sua probabilidade de estarem conectados.

Foi neste momento, que o grupo de pesquisa observou como as pessoas computam distâncias sociais de uma forma bastante complexa, geralmente caracterizando uma possível violação da condição formal matemática conhecida como *desigualdade triangular*. Segundo essa desigualdade, o comprimento de qualquer lado de um triângulo é sempre menor ou igual à soma dos comprimentos dos outros dois lados. Na Figura 2.23, a condição da desigualdade triangular garante que:

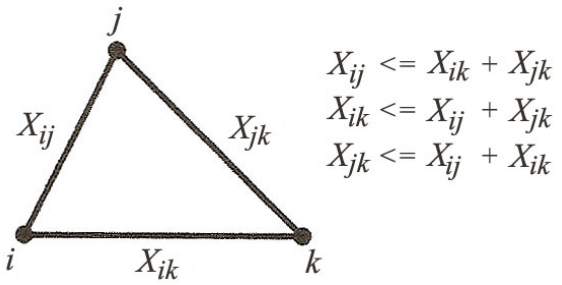


Figura 2.23: *Desigualdade Triangular.*

Para entender como a violação dessa condição acontece, é necessário entender inicialmente como distâncias podem ser medidas nesse contexto. A primeira forma é a distância através da rede, ou o número de laços do menor caminho capaz de conectar dois pontos. No caso dos pontos i e j da Figura 2.23, este caminho seria X_{ij}.

Mas essa não é a definição de distância que as pessoas usam normalmente quando pensam sobre quão distantes estão de alguém. Os indivíduos tendem a se identificar em termos de contextos, fazendo uso de múltiplas dimensões. A distância geográfica é um exemplo óbvio, mas profissão, educação, religião e interesses pessoais, também influem nas avaliações de proximidade das pessoas [107].

Com essa perspectiva, os pesquisadores perceberam que estar próximo em uma determinada dimensão não implica necessariamente proximidade em outra. Obviamente, se duas pessoas estão próximas em apenas uma dimensão, podem se considerar próximas em um sentido absoluto, mesmo que estejam bastante distantes em outras dimensões [107]. As pessoas só precisam ter um contexto de interação, o que pode ser suficiente para que elas se autoconheçam.

Como exemplificado na Figura 2.24 [107], os mesmos indivíduos i e k da Figura 2.23, podem se perceber ambos como próximos de j, onde i está próximo em uma determinada dimensão d, por exemplo, geografia, e k em outra, por exemplo, ocupação. Como apenas as menores distâncias contam, não importa que j esteja bem distante tanto de i quanto de k em outros aspectos. Mas, como i e k estão distantes em ambas as dimensões, percebem-se mutuamente como distantes [107].

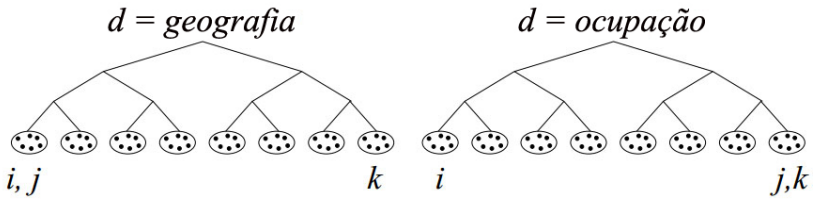


Figura 2.24: *Multidimensões.*

Essa situação geralmente ocorre quando uma pessoa tem dois amigos que foram conhecidos em circunstâncias diferentes, sem que os mesmos possuam algo em comum, o que define a situação de um *buraco estrutural*. Mas, o fato é que eles possuem algo em comum, a pessoa que os une, deixando-os próximos mesmo que de forma inconsciente.

Como concluíram os pesquisadores, a identidade social exibe uma natureza multidimensional. Indivíduos em diferentes contextos sociais podem explicar a violação da desigualdade triangular. A Figura 2.24 mostra que o nó *i* reconhece o nó *k* como distante, pois inexiste uma dimensão em que ambos estejam presentes e se autorreconheçam. Desta forma, apesar da condição da desigualdade triangular garantir que o caminho (*i*,*k*) seja menor que o caminho (*i*,*j*,*k*), por esta perspectiva, é exatamente o oposto que acontece.

Como o nó *i* está intimamente ligado ao nó *j*, e o nó *j* está intimamente ligado ao nó *k*, o que representa curtas distâncias entre os nós, a distância a ser percorrida de *i* até *k*, é menor se percorrida pelo caminho (*i*,*j*,*k*). Para Watts, a natureza multidimensional da identidade social é "o que encurta as distâncias entre as pessoas nas redes sociais" [107].

Consequentemente, quanto mais contextos as pessoas compartilham, mais próximas estão, e, portanto, mais chances possuem de se conectar. Mesmo quando suas associações sejam altamente homófilas, ao pertencer a determinados grupos, indivíduos adquirem características que lhes dão mais probabilidade de interagir entre si.

Dentro dessa perspectiva, Watts concluiu no projeto que em vez de começar com uma noção de distância para construir os grupos, seria interessante começar com os grupos para definir uma medida de distância [106]. Em vez de indivíduos que se escolhem diretamente, esses indivíduos poderiam optar por juntar-se a certo número de grupos.

A técnica que ajudou Watts a definir o conceito de distância em função da estrutura de grupo foi expressá-lo em uma *rede por afiliação* [106]. Dois nós neste tipo de rede podem ser considerados afiliados se participam do mesmo grupo. Para Watts, sem afiliações, a chance de duas pessoas se conectarem se torna desprezível, e quanto mais afiliações elas tiverem, maior a probabilidade de que interajam como amigos ou conhecidos. Os grupos, por sua vez, também podem ser afiliados em virtude de membros em comum, o que permite que eles se sobreponham ou apresentem intersecção.

Como redes por afiliação consistem em dois tipos distintos de nós, os mesmos foram denominados pelo pesquisador de *atores* e *grupos*, e representados através de uma *rede bipartida* [106]. Observa-se, através do painel central da Figura 2.25 [106], que em

uma rede bipartida, os dois conjuntos de nós são representados separadamente e apenas nós de tipos diferentes podem ser conectados, criando relações de pertencimento.

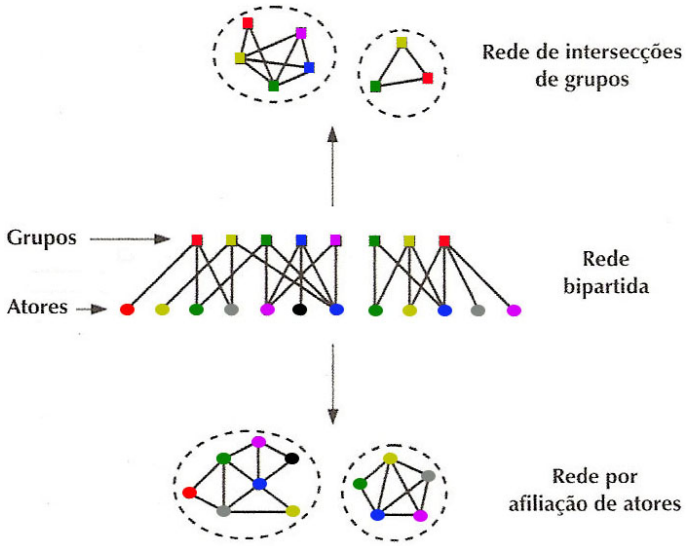


Figura 2.25: *Rede Bipartida.*

As redes *unipartidas* são representações das relações observadas nos dados mensuráveis da rede, como uma lista de quem conhece quem. Todavia, este tipo de dado não pode precisar de onde vêm estas relações [106]. Ao esboçarem uma representação da estrutura social, com a representação bipartida completa, o grupo de pesquisa conseguiu verificar a origem das relações interpessoais, apresentando a estrutura das redes por afiliação de atores e de intersecção de grupos simultaneamente.

Tomando as duas distribuições da rede bipartida, grupos por ator e atores por grupo, os pesquisadores assumiram que a correlação entre atores e grupos ocorresse de forma aleatória. Obviamente, esta dinâmica não procede no mundo real, mas como haviam definido em modelos anteriores, os cientistas esperavam que as decisões individuais de escolha de grupos fossem completamente imprevisíveis, tornando-se impossível distingui-las da mera aleatoriedade.

Redes aleatórias não possuem aglomeração, o que, como característica, tornou a representação bipartida das redes por afiliação de extrema utilidade. Os pesquisadores demonstraram que mesmo uma rede aleatória bipartida, sem propriedades estruturais, será aglomerada. Por outro lado, a aleatoriedade garante que estas redes fiquem altamente conectadas e apresentem caminhos globais curtos. Portanto, representando o problema no que parecia uma forma sociologicamente plausível, os pesquisadores concluíram que redes por afiliação aleatórias serão sempre redes de mundo pequeno. Elas consistem de conjuntos superpostos, ligados entre si pela coparticipação de indivíduos em múltiplos grupos, algo que remete à imagem de sociedade visualizada por Granovetter.

CAPÍTULO 3

Algoritmos de Busca de Caminhos

Informalmente, um algoritmo pode ser definido como uma sequência de ações executáveis para a obtenção de uma solução referente a determinado tipo de problema. De acordo com Dijkstra, um algoritmo corresponde a um padrão de comportamento particularmente descrito e expresso em termos de um conjunto finito de ações [38].

Embora algoritmos tenham uma longa história em matemática, sua noção propriamente dita não estava definida precisamente até o século XX [92]. Antes disso, os matemáticos somente possuíam uma noção intuitiva sobre os mesmos, através da qual passavam a utilizá-los e a descrevê-los.

A definição formal só se concretizou nos artigos de 1936 de Alonzo Church e Alan Turing [92]. Apesar da utilização de técnicas distintas por parte de ambos os pesquisadores, as duas definições foram demonstradas ser equivalentes. Para definir algoritmos, Church usou um sistema notacional chamado $\lambda-$ *cálculo*, enquanto Turing o fez com suas *máquinas*, configurando a *tese de Church-Turing* [92].

Conforme apontou Knuth, na área de análise de algoritmos, existem dois tipos de problemas a serem observados [63]:

- **Análise de um algoritmo particular:** Refere-se ao custo de execução e quantidade de memória utilizada por um algoritmo na resolução de um problema;
- **Análise de uma classe de algoritmos:** Compreende a análise de uma família de algoritmos, a fim de identificar o de menor custo para resolver um problema.

O projeto de algoritmos é fortemente influenciado pelo estudo de seus comportamentos. Depois que um problema é analisado e decisões de projeto são finalizadas, o algoritmo tem que ser implementado. É neste momento que o projetista estuda as várias opções de algoritmos a serem utilizadas, cujos aspectos de tempo de execução e espaço ocupado são relevantes [111]. Muitos destes algoritmos são encontrados em áreas como pesquisa operacional, otimização, teoria dos grafos, entre outras, sendo estas citadas, fundamentais para a concepção das técnicas de busca implementadas nesta obra.

Os projetistas de *software* utilizam a crescente capacidade dos computadores para resolver problemas cada vez mais complexos. Todavia, os problemas a serem

resolvidos pelos computadores, em última análise, não podem depender do seu poder computacional, o que torna fundamental uma comparação da eficiência dos algoritmos.

Nesse contexto, a complexidade dos problemas reais levou ao surgimento de abordagens díspares, como o paradigma biológico ou conexionista, que se apresentam como uma alternativa ao paradigma computacional simbólico, para a resolução eficiente destes problemas.

É por esta razão, que os avanços no estudo de algoritmos são capitais para o aprimoramento de técnicas computacionais. As últimas décadas se caracterizaram pela crescente produção de trabalhos importantes nesta área, que se habilitaram a desenvolver soluções com diferentes abordagens para os mais variados problemas, em especial para o que se relaciona com a criação de técnicas de busca, fundamental nesta dissertação: o *problema do caminho mínimo*.

3.1 Problema do Caminho Mínimo

Um dos problemas reais mais estudados em otimização, que vem recebendo recentemente grande atenção da comunidade científica, é o *problema do caminho mínimo*. Com um papel essencial nesta dissertação, este problema consiste genericamente, em encontrar o caminho de menor custo entre dois nós na rede, considerando a soma dos custos associados às arestas percorridas. Os algoritmos especializados em solucionar este tipo de problema recebem a denominação de *algoritmos de busca de caminhos*.

Formalmente, em um problema do caminho mínimo, tem-se um grafo ponderado $G = (V,E)$, com função peso $w : E \rightarrow \mathbf{R}$ mapeando arestas para pesos de valores reais. O peso do caminho $p = (v_0, v_1, ..., v_k)$ é a soma dos pesos de suas arestas constituintes [32].

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Define-se o peso do caminho mais curto de u até v por [32]:

$$\delta(u,v) = \begin{cases} \min\{ w(p) : u \xrightarrow{p} v & \text{se existe um caminho de u até v,} \\ \infty & \text{em caso contrário.} \end{cases}$$

Um caminho mínimo desde o vértice u até o vértice v é então definido como qualquer caminho p com peso $w(p) = \delta(u,v)$. O problema apresenta subestrutura ótima, pois qualquer subcaminho de um caminho mínimo, também é um caminho mínimo.

A título de exemplificação, têm-se o grafo representado na Figura 3.1, correspondente ao mapeamento da distância em *km* entre oito locais específicos de uma cidade. O caminho mínimo entre os nós S e T, assinalado na imagem com traços pontilhados, corresponde ao caminho $(S-3-6-5-4-T)$, totalizando um *peso* de 10 *km*. Este seria o menor trajeto possível entre os dois pontos determinados.

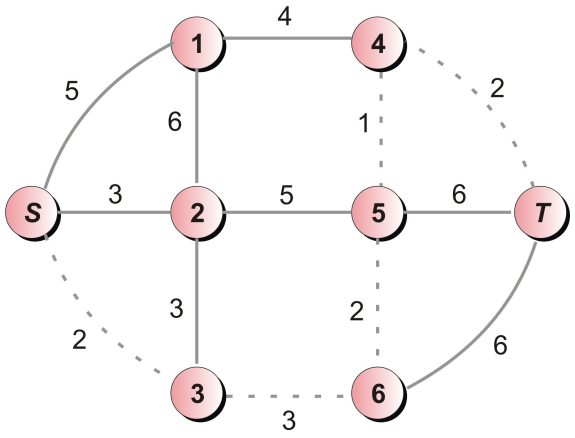


Figura 3.1: *Grafo da Distância Entre Oito Locais de uma Cidade.*

Basicamente, são quatro as variantes do problema do caminho mínimo [32]:

- **Problema de único destino:** consiste em determinar o menor caminho entre cada um dos nós do grafo e um determinado nó de destino;
- **Problema de única origem:** consiste em determinar o menor caminho entre um determinado nó e todos os outros nós do grafo;
- **Problema de origem destino:** consiste em determinar o menor caminho entre dois nós específicos, como apresentado na ilustração anterior. Esta foi a modalidade analisada na dissertação para a criação do mecanismo de busca social;
- **Problema de todos os pares:** consiste em determinar o menor caminho entre todos os pares de nós presentes no grafo.

O problema do caminho mínimo se adapta a diversas situações práticas, que de alguma forma possuam grandezas como distância, tempo ou despesas, acumuladas linearmente ao longo do percurso da rede. Em redes de computadores, por exemplo, os nós podem representar equipamentos diversos, as arestas correspondem a trechos de cabeamento, e os custos ou pesos poderão estar associados à taxa máxima de transmissão de dados. Neste caso, a solução do problema seria a rota de transmissão mais rápida.

Uma das possibilidades de se resolver esse tipo de problema é calcular a distância de todas as rotas e escolher a menor encontrada. Este raciocínio usa o famoso método de "força bruta" ou de "busca exaustiva", o que acaba requerendo muito poder computacional para resolver o problema [32]. Este método resulta em um número de opções igual a $n(n-1)(n-2)...(2)(1) = n!$. Se houver 100 nós interligados, existem 10^{158} opções de rota. Caso fosse possível testar um bilhão de soluções por segundo, seria necessário um tempo igual a 10^{141} anos para encontrar a melhor solução [11]. Pode-se perceber que por esta perspectiva o problema se torna intratável e, tendo em vista sua importância, são necessárias outras técnicas capazes de resolvê-lo.

Existem basicamente dois tipos de estrutura de algoritmos de busca para o cálculo de caminhos mínimos: árvores e matrizes [25]. Os algoritmos em árvore determinam os caminhos mínimos de um nó para todos os outros nós da rede ou simplesmente entre um par de nós. Neste tipo de algoritmo, a solução é obtida construindo-se passo a passo uma árvore de caminhos mínimos. Já os algoritmos com estrutura de matriz são capazes de obter os caminhos mínimos entre todos os pares de nós da rede, sendo estes caminhos mínimos obtidos simultaneamente.

Na literatura, existem alguns algoritmos de busca capazes de resolver o problema de forma exata. Dentre estes, os mais comumente utilizados com estrutura em árvore são *Dijkstra* e *Ford/Moore*, e com estrutura em matriz: *Floyd*, *Dantzig* e *Double Sweep* [25].

Se alguma dessas técnicas for capaz de resolver o problema de forma eficiente, o mais indicado é que sejam utilizadas. Entretanto, ao encontrar um problema intratável, uma boa solução é utilizar técnicas de aproximação.

Existem outros algoritmos propostos que buscam minimizar o tempo gasto nesse tipo de processamento, utilizando-se de abordagens inspiradas nos processos evolutivos e orgânicos da natureza, tais como os *Algoritmos Genéticos* ou a *Otimização por Colônia de Formigas*. Estes últimos, assim como o Algoritmo de Dijkstra, serão objetos de estudo deste trabalho, e, portanto, merecem o destaque apropriado.

3.2 Algoritmo de Dijkstra

O Algoritmo de Dijkstra, concebido pelo cientista da computação holandês Edsger Dijkstra em 1956 e publicado em 1959, soluciona o problema do caminho mínimo em um grafo dirigido ou não dirigido. Este algoritmo é utilizado para determinar o menor caminho de um nó para outro nó ou para todos os outros nós da rede, desde que as arestas contenham pesos não negativos [32].

Essa característica valorada das arestas ou dos arcos é exatamente o diferencial do Algoritmo de Dijkstra perante o algoritmo *BFS - Breadth First Search* [32], ou técnica de busca em largura, também capaz de localizar o caminho mínimo entre dois pontos, desde que os grafos não sejam ponderados. O que permite esta capacidade ao Algoritmo de Dijkstra é o tipo de estrutura de dados utilizado pelo mesmo. Ao invés de trabalhar com um fila *FIFO* inerente ao *BFS*, o Algoritmo de Dijkstra percorre uma fila de prioridades.

Possuindo uma estratégia "gulosa", o Algoritmo de Dijkstra sempre toma uma decisão que parece ser a melhor no momento. Para a Teoria dos Grafos, esta estratégia se torna altamente conveniente, já que todo subcaminho de um caminho mínimo também é um menor caminho entre dois vértices. Desta forma, constrói-se o menor caminho entre dois pontos no grafo determinando todos os menores caminhos intermediários.

Os nós são divididos em dois grupos: os que já tiveram os caminhos mais curtos determinados, S, e os candidatos ou de fronteira, Q, que ainda estão na fila de prioridade, correspondentes ao conjunto $V - S$. O conjunto S é inicializado para conter apenas o nó de origem. Os vizinhos imediatos da origem são colocados no conjunto Q, sendo registrados os custos para poder atingi-los a partir da mesma.

A cada passo do algoritmo, os nós de Q são verificados para determinar qual seria a melhor opção para expandir a pesquisa. Será escolhido e transferido para S aquele nó cujo custo acumulado seja o menor dentre os candidatos, e consequentemente todos os vizinhos deste nó serão então adicionados ao conjunto Q [35]. A pesquisa para quando o nó de destino for alcançado, ou quando não houver mais nós a serem percorridos, o que implica em inexistência de caminho entre os nós de origem e destino. Trata-se, portanto, de um algoritmo iterativo que se utiliza da seguinte fórmula de recorrência [25]:

$$d(v)^i = \left\{ \min \{ d\ (v)^{i-1}, d(u) + d(u,v) \} \right\}$$

onde: $d(v)^i$ corresponde ao tamanho do caminho da origem s até o nó v na iteração corrente, $d(u)$ corresponde ao tamanho do caminho da origem s até o nó u, e $d(u,v)$ corresponde ao tamanho do arco (u,v).

Esse processo de minimização, apresentado na fórmula de recorrência do algoritmo, é denominado de *relaxamento* [32]. Esta técnica consiste em testar se é possível melhorar o caminho mais curto para v encontrado até o momento, pela passagem através de u, e em caso afirmativo, atualizar a respectiva distância e o nó predecessor. A demonstração pode ser visualizada na Figura 3.2 [32].

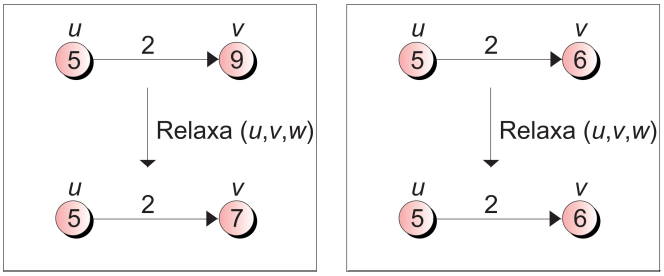


Figura 3.2: *Relaxamento da Aresta (u,v).*

Em ambos os casos apresentados no exemplo acima, a estimativa de caminhos mínimos de cada vértice é mostrada em seu interior, sendo o peso de $w(u,v) = 2$. Na situação do painel esquerdo como $d(v)$ é maior que $d(u) + w(u,v)$ antes do relaxamento, o valor de $d(v)$ diminui. Já na situação do painel direito, $d(v) <= d(u) + w(u,v)$ antes da etapa de relaxamento e, assim não tem seu valor alterado pela técnica.

De forma genérica o algoritmo 3.1 apresenta o Algoritmo de Dijkstra.

Algoritmo 3.1: $Dijkstra(G, w, s, t)$

Entrada: grafo $G(V, E)$, vetor de pesos não negativos das arestas w e vértices de origem s e destino t.

Saída: vetor de distância $d(v)$ e vetor de predecessores $\pi(v)$.

```
1  para cada v ∈ V faça
2      d[v] ← ∞
3      π(v) ← Null
4  fim
5  d[s] ← 0
6  S ← {}
7  Q ← V
8  enquanto |Q| ≠ 0 faça
9      u ← extrairMinimo(Q)
10     S ← S ∪ {u}
11     para cada v ∈ Adj[u] faça
12         se d[v] > (d[u] + w(u,v)) então
13             d[v] ← d[u] + w(u,v)
14             π(v) ← u
15         fim
16     fim
17 fim
```

A função $extrairMinimo(Q)$ utilizada na linha 9 do algoritmo 3.1 é responsável por escolher o nó cujo custo acumulado seja o menor dentre os candidatos. O código entre as linhas 11 a 16 realiza a técnica do relaxamento citada anteriormente.

O Algoritmo de Dijkstra vem sendo aperfeiçoado desde sua criação através do uso de estruturas de dados mais eficientes. O tempo de execução do algoritmo depende de como a fila de prioridade Q é implementada. Quando o grafo é suficientemente esparso, é interessante implementar a fila de prioridade através de um *heap mínimo binário*, o que diminui a complexidade computacional do algoritmo de $O(V^2)$ para $O(V \lg V)$ [32].

Em ciência da computação, um *heap binário* é uma estrutura de dados organizada como árvore binária balanceada. Existem dois tipos de *heap binários*, o *heap máximo*, caracterizado por cada nó na árvore possuir chave maior que todos os seus filhos, e o *heap mínimo*, caracterizado exatamente pelo inverso [32].

O *heap mínimo* é implementado no Algoritmo de Dijkstra possuindo como chave os valores de distância do vértice fonte até os nós correspondentes. À medida que esta distância diminui, as prioridades dos respectivos nós aumentam, deixando-os cada vez mais próximos de serem processados. A Figura 3.3 ilustra um *heap mínimo binário*:

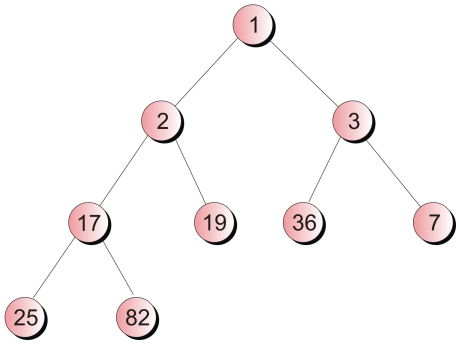


Figura 3.3: *Heap Mínimo.*

A sequência de diagramas das Figuras 3.4 e 3.5 ilustra o funcionamento do Algoritmo de Dijkstra, detectando o menor caminho entre o nó 1 e todos os demais [70]:

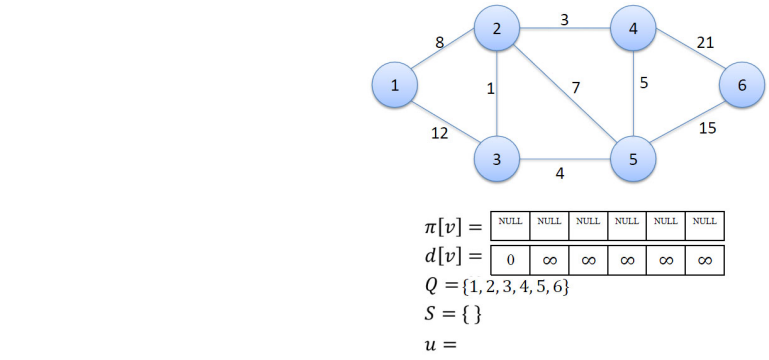


Figura 3.4: *Inicialização do Algoritmo de Dijkstra.*

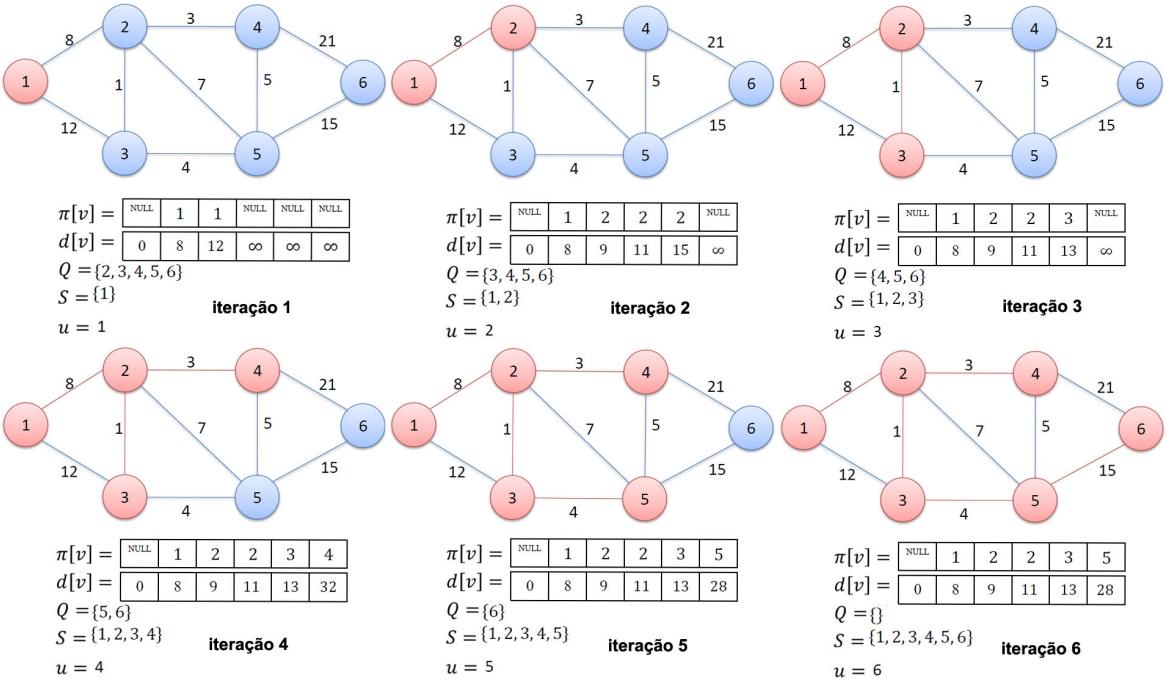


Figura 3.5: *Iterações do Algoritmo de Dijkstra.*

3.3 Otimização por Colônia de Formigas

No fim da década de quarenta do século XX, o francês entomologista Pierre Paul Grassé observou que algumas espécies de cupins reagiam ao que ele chamou de estímulos importantes [52]. Ele percebeu que os efeitos de tais reações podem atuar como novos estímulos relevantes tanto para os insetos que os produziu quanto para o restante da colônia. Grassé utilizou o termo *estigmergia* para descrever este tipo particular de comunicação em que os trabalhadores são estimulados pelo desempenho alcançado [53].

O comportamento de coleta de alimentos de muitas sociedades de formigas baseia-se exatamente nesse tipo de comunicação, que de fato consiste em uma troca indireta de mensagens através de uma substância química volátil denominada *feromônio* [36]. Enquanto caminham do formigueiro para as fontes de alimento e vice-versa, as formigas depositam feromônio na superfície, formando uma trilha química.

Em uma experiência realizada pelo grupo de pesquisa do cientista Jean-Louis Deneubourg, conhecida como a *ponte binária* [36] [49], um ninho de uma colônia de formigas argentinas foi conectado a uma fonte de alimento por duas pontes de comprimentos diferentes, como observado na Figura 3.6 [28].

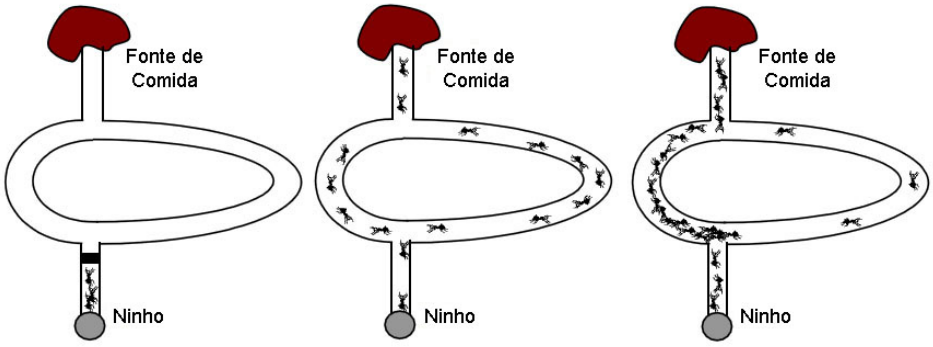


Figura 3.6: *Experimento da Ponte Binária.*

Inicialmente, cada formiga escolhia aleatoriamente uma das duas pontes através da qual alcançavam a fonte de comida, depositando feromônio ao longo de sua trajetória *ninho-comida-ninho*. Com o tempo, as flutuações estocásticas da escolha inicial tornavam-se reduzidas e um segundo mecanismo desempenhava um papel importante.

Como as formigas que escolheram o menor caminho faziam o percurso mais rapidamente que as outras, este caminho acabava recebendo uma maior quantidade de feromônio em relação ao outro, em um mesmo intervalo de tempo. O resultado deste processo de retroalimentação conhecido por *feedback positivo* é que depois de algum tempo, a colônia inteira convergia no sentido da utilização da mesma ponte [36].

A ideia básica, é que se em um determinado ponto uma formiga tem de escolher entre caminhos diferentes, aqueles que foram fortemente escolhidos por formigas anteriores, possuirão elevado nível de feromônio, e serão escolhidos com maior probabilidade.

A grande importância dessa descoberta, é que altos níveis de feromônio são sinônimo de caminhos curtos [49]. Dessa forma, mesmo sem grandes propriedades visuais, este comportamento simples de seguir trilhas possibilita às formigas a emergência de um comportamento mais complexo: encontrar caminhos mínimos entre dois pontos.

Além disso, as formigas podem se adaptar a alterações no meio ambiente. Por exemplo, elas são capazes de encontrar um novo caminho mínimo, caso o antigo não seja mais viável devido ao surgimento de um obstáculo, situação ilustrada na Figura 3.7 [41].

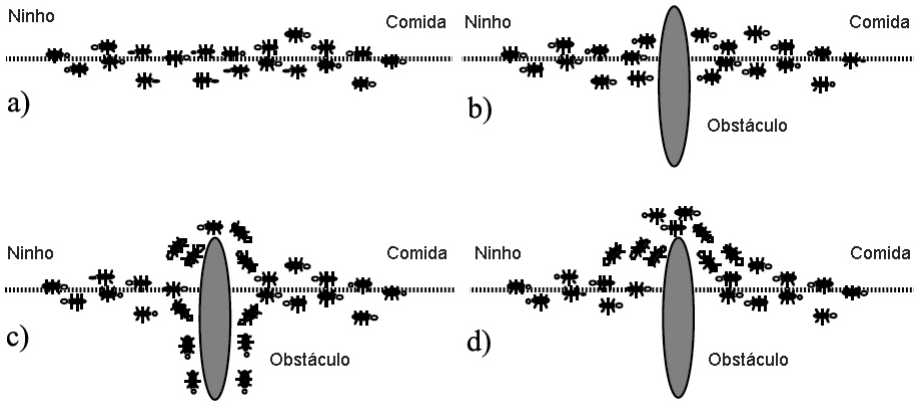


Figura 3.7: *Obstáculo Sendo Superado Pelas Formigas.*

A fim de formalizar todo o processo forrageador criado pelas formigas, Marco Dorigo propôs o seguinte modelo matemático apresentado na Figura 3.8 [43]. As distâncias entre os pontos D e H, entre B e H, e entre B e D via C são iguais a 1, e o ponto C está posicionado a meio caminho entre D e B, como ilustrado na posição (a) da Figura 3.8.

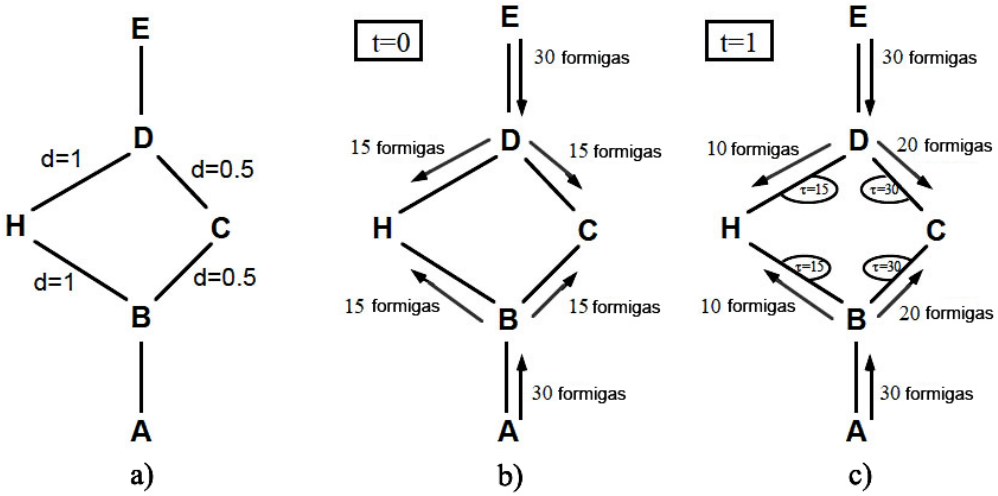


Figura 3.8: *Processo Forrageador das Formigas.*

Existem 30 novas formigas vindo até o ponto B a partir de A, e 30 indo até D a partir de E. Cada formiga percorre a distância $d = 1$ em uma unidade de tempo, depositando durante este percurso, um rastro de feromônio com intensidade 1.

Quando $t = 0$ não há trilha ainda, mas 30 formigas estão em B e 30 em D. A escolha sobre qual caminho percorrer é completamente aleatória. Portanto, em média, 15 formigas de cada nó irão para H e 15 para C, como ilustrado na posição (b) da Figura 3.8.

Em $t = 1$, as 30 novas formigas que vêm para B de A encontram um rastro de feromônio de intensidade 15 sobre o caminho que leva à H, depositado pelas 15 formigas que vieram de B. Não obstante, elas também farejam um rastro de feromônio de intensidade 30 no caminho para C, obtido com a soma da trilha estabelecida pelas 15 formigas que vieram de B e pelas 15 formigas que já atingiram B proveniente de D via C, situação ilustrada na posição (c) da Figura 3.8.

Como a distância do lado direito do percurso (B,C,D) ou (D,C,B) é a metade do seu lado esquerdo (B,H,D) ou (D,H,B), em $t = 1$, as formigas que optaram pelo lado menor atingiram o ponto final do percurso, enquanto que as outras ainda estão na metade do caminho. Dessa forma, este lado menor possuirá o dobro do feromônio depositado no lado maior, tornando tendenciosa a probabilidade de escolha de um dos caminhos, por parte das novas formigas. Consequentemente, o número esperado de novas formigas que vão para C e H, em $t = 1$, será 20 versus 10, respectivamente [43], o mesmo ocorrendo com as 30 novas formigas em D, que vieram a partir de E.

A pesquisa realizada pelo grupo de Deneubourg foi a principal fonte de inspiração para o desenvolvimento de algoritmos baseados no comportamento forrageador das formigas. O primeiro algoritmo criado, alicerçado nestes conceitos, foi proposto por Dorigo no início dos anos noventa, sendo denominado como *Ant System - (AS)* [43].

O *AS* utiliza-se de formigas artificiais que constroem soluções para o problema de otimização considerado. Estes seres trocam informações sobre a qualidade de suas soluções através de um esquema de comunicação similar ao adotado por formigas reais.

Embora não exista uma estrutura centralizada de controle que estabelece como as formigas artificiais devem se comportar, e mesmo não havendo um modelo explícito do ambiente, as interações locais entre as mesmas geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema.

Esse algoritmo foi aplicado pela primeira vez ao *problema do caixeiro viajante* [92], sendo recentemente modificado, tanto para melhorar o seu desempenho quanto para aplicá-lo a outros problemas de otimização, como o problema do caminho mínimo. Dentre as versões destacam-se: *Ant Colony System (ACS)*, *Max-Min Ant System*, e *ASrank*. Estas novas versões, bem como a original, obtiveram resultados para alguns problemas de otimização que estão entre as melhores heurísticas disponíveis atualmente [40].

A metaheurística *Ant Colony Optimization (ACO)* foi o resultado de um esforço realizado por Dorigo para definir um quadro comum, ou *framework*, para todas as versões do *AS* [40], proporcionando uma visão unitária da investigação em curso neste campo.

Dessa forma, a *ACO* se tornou uma técnica robusta inerente às diversas vertentes

de pesquisa sobre a *Inteligência de Enxames*, ou *Swarm Intelligence*, que é hoje um campo perfeitamente desenvolvido da Inteligência Artificial.

Segundo essa metaheurística, as formigas artificiais constroem soluções de forma probabilística utilizando duas informações. A trilha de feromônio artificial, que muda dinamicamente durante a execução do programa de modo a refletir a experiência adquirida, e a informação heurística específica do problema a ser resolvido [40].

De acordo com o algoritmo original, *AS*, a probabilidade da formiga k que está no vértice i escolher o vértice j é dada pela seguinte regra de transição [43]:

$$p_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(n_{ij})^{\beta}}{\sum_{l \in n_i^k}(\tau_{il})^{\alpha}(n_{il})^{\beta}}$$

onde: τ_{ij} é feromônio da aresta (i, j), α e β são parâmetros que determinam a influência do feromônio e da informação heurística, e n_i^k é a vizinhança factível da formiga, ou o conjunto de vértices l não visitados pela formiga k, posicionada em i.

Também está associado à aresta (i, j) um valor heurístico n_{ij} dado por $\frac{1}{d_{ij}}$, representando a atratividade da mesma. O valor n_{ij} é inversamente proporcional a distância d_{ij} entre os vértices i e j, contribuindo para que os caminhos curtos sejam melhor avaliados.

No feromônio τ_{ij} ocorrem particularmente dois eventos. A *evaporação*, que evita que o mesmo se acumule e cresça indefinidamente, além de favorecer o esquecimento de decisões ruins tomadas no passado da busca, e o *depósito* de feromônio das formigas que passaram sobre (i, j). Dessa forma, após todas as formigas construírem suas viagens, o feromônio τ_{ij} é atualizado pela seguinte fórmula:

$$\tau_{ij} = (1 - \rho)_{\tau_{ij}} + \sum_{k=1}^{m} \Delta\tau_{ij}^{(k)}$$

onde: $0 < \rho <= 1$ é a taxa de evaporação de feromônio.

A expressão $\Delta\tau_{ij}^{(k)}$ refere-se à quantidade de feromônio que a formiga k deposita sobre a aresta (i, j), sendo seu cálculo obtido por:

$$\Delta\tau_{ij}^{(k)} = \begin{cases} \frac{1}{L_k} & \text{se a aresta } (i, j) \text{ pertence a viagem } S_k, \\ 0 & \text{em caso contrário.} \end{cases}$$

onde: L_k representa o comprimento do caminho construído pela formiga k.

O critério de parada do algoritmo é estabelecido entre um número máximo de iterações ou quando fica evidente sua estagnação, devido à situação na qual todas as formigas seguem sempre o mesmo percurso. A complexidade do *AS* foi calculada por Dorigo como $O(NCn^2m)$ [43], onde n é o número de vértices, m, o número de formigas, e NC, o número máximo de iterações ou ciclos definidos para o algoritmo. Caso, experimentalmente, possa ser encontrada uma relação linear entre o número de vértices e o melhor número de formigas, a complexidade do algoritmo será $O(NCn^3)$ [43].

Dentre todas as variações desse algoritmo, a que mais tem sido utilizada por suas grandes contribuições foi o $AntColonySystem(ACS)$, proposto por Dorigo em 1996 [42].

O *ACS* difere do *AS* anterior por causa de três aspectos principais [42]:

- a regra de transição fornece uma maneira direta de equilíbrio entre a exploração de novas arestas e a explotação do conhecimento acumulado sobre o problema;
- a regra de atualização global, ou atualização *offline* de feromônio, é aplicada apenas para as arestas que pertencem ao melhor caminho das formigas;
- enquanto as formigas constroem uma solução, uma regra de atualização local, ou atualização *online* de feromônio, é aplicada.

No algoritmo *ACS*, a regra de transição utilizada pelas formigas durante o processo de construção foi modificada para uma regra de transição *pseudorandômica*. A probabilidade de uma formiga se deslocar do vértice i para o vértice j depende de uma variável q aleatória uniformemente distribuída sobre [0,1], e um parâmetro q_0 também escolhido aleatoriamente entre [0,1]. Se $q <= q_0$, então a escolha do vértice j é dada por:

$$j = arg\max_{l \in n_i^k} \left\{ (\tau_{il})^{\alpha} (n_{il})^{\beta} \right\}$$

Caso $q > q_0$, então é utilizada a regra de transição do algoritmo *AS*.

A segunda mudança expressiva do *ACS* em relação ao *AS*, é que apenas a melhor formiga, ou a formiga que encontrou o menor caminho, deposita feromônio após cada iteração, ao invés de todas as formigas que completaram um percurso válido. Esta atualização de feromônio *offline* é obtida por:

$$\tau_{ij} = (1-\rho)_{\tau_{ij}} + \rho \Delta \tau_{ij}$$

somente se (i, j) pertencer ao caminho da melhor formiga.

A atualização *online* de feromônio é a última alteração implementada no *ACS*. Esta técnica é realizada por todas as formigas após cada etapa de construção, sendo então aplicada em todas as arestas percorridas. Este cálculo é realizado pela seguinte fórmula:

$$\tau_{ij} = (1-\varphi)_{\tau_{ij}} + \varphi \tau_0$$

onde: $0 < \varphi <= 1$ é o coeficiente de decaimento de feromônio, e τ_0 é o valor inicial do feromônio.

O principal objetivo da atualização *online* é diversificar a pesquisa realizada por formigas subsequentes durante uma iteração, através da diminuição da concentração de feromônio nas arestas percorridas.

A complexidade do *ACS* foi calculada por Dorigo como $O(n^2t)$, onde n é o número de vértices e t, o número de *tours*, ou caminhos gerados pelas formigas [41].

O algoritmo 3.2 apresenta o *Algoritmo de Otimização por Colônia de Formigas (ACO)* em sua forma genérica.

Algoritmo 3.2: $ACO(G,w,s,t)$

Entrada: grafo $G(V,E)$, vetor de pesos não negativos das arestas w e vértices de origem s e destino t.

Saída: vetor de vértices S^*, representando o caminho da melhor formiga.

```
1  coloque as formigas no vértice de origem
2  inicializa os feromônios das arestas
3  L* ← ∞
4  para t = 1 até número máximo de iterações faça
5      para k = 1 até m faça
6          enquanto a formiga k não construir a viagem S_k faça
7              selecione o próximo vértice pela regra p_ij^k
8              se atualização online de feromônio então
9                  atualize feromônio da aresta (i, j)
10             fim
11         fim
12         calcule a distância L_k da viagem S_k
13         se L_k < L* então
14             S* = S_k
15             L* = L_k
16         fim
17     fim
18     se atualização offline de feromônio elitista então
19         atualize feromônio das arestas de S*
20     senão
21         atualize feromônio das arestas de todas S_k
22     fim
23 fim
```

3.4 Algoritmos Genéticos

Em 1958, o pesquisador Oliver Selfridge, interessado no estudo sobre como os sistemas se modificam, evoluem e aprendem, apresentou um trabalho intitulado como *Pandemômio: Um Paradigma para Aprendizagem* [60]. Sua inspiração foi a pesquisa de

Alan Turing sobre o mistério da *morfogênese*, ou a capacidade de inúmeras formas de vida de desenvolverem corpos mais elaborados a partir de inícios incrivelmente simples.

Na realidade, o *Pandemômio* consistia mais em um modo de abordar um problema do que em um *software* específico. Como os recursos computacionais eram precários na época, o objetivo do pesquisador era ensinar um computador a reconhecer padrões mal definidos ou inconstantes, como o som da linguagem falada.

Consequentemente, Selfridge criou uma quantidade de miniprogramas limitados, aos quais denominou de *demônios*. Sua ideia consistia em ter um conjunto destes demônios agindo ao longo de uma hierarquia, cooperando na resolução de problemas complexos por intermédio do simples conhecimento inerente a cada elemento.

Segundo Selfridge o esquema esboçado era uma seleção natural dos *demônios* processadores. Se eles servissem a uma função útil, sobreviveriam, e talvez fossem a fonte para outros *subdemônios* que seriam julgados por seus próprios méritos [60].

Na década de 60, John Holland estava disposto a fazer com que a lenta máquina calculadora *701 da IBM* aprendesse de modo mais orgânico, *bottom-up*, não diferente do *Pandemônio* de Selfridge [60]. Como Turing, Holland queria explorar o modo pelo qual regras simples podiam levar a comportamentos complexos; como Selfridge, ele queria criar um *software* que fosse capaz de aprendizado ilimitado.

O maior avanço de Holland foi controlar o poder de outro sistema aberto e *bottom-up*: a seleção natural. Construindo sobre o modelo do *Pandemônio*, ele tomou a lógica da evolução darwiniana e da genética de Mendel e as transformou em código, nomeando sua criação de *Algoritmo Genético (AG) [60]*.

O Algoritmo Genético é uma metaheurísitca de pesquisa estocástica que apresenta a evolução biológica como uma técnica para resolução de problemas de otimização. Os indivíduos gerados por este algoritmo evoluem para encontrar a solução de um dado problema, sendo cada um deles uma representação possível da solução procurada [12].

Com a dinâmica do algoritmo, esses indivíduos competem entre si, e os mais aptos, em detrimento dos candidatos mais fracos, são selecionados para serem cruzados em uma próxima geração, como reza o princípio da teoria da evolução das espécies de Darwin. Consequentemente, a cada nova geração, devem surgir indivíduos mais adaptados, implicando em uma convergência natural dos mesmos para a solução do problema. A aptidão média da população aumenta a cada iteração, e a repetição deste processo inúmeras vezes permite a descoberta de melhores resultados.

Algoritmos Genéticos são estudados em vários campos de pesquisa, fornecendo métodos alternativos para a resolução de problemas complexos, geralmente pertencentes à área de otimização. Dentre as principais características do algoritmo destacam-se [64]:

- São paralelos por natureza, explorando o espaço de solução em várias direções simultaneamente;

- Funcionam bem em problemas com propriedades dinâmicas, ou que possuem muitos ótimos locais;

- Não requerem conhecimento especializado sobre o problema.

Por seus diferenciais comprovados, recentemente Algoritmos Genéticos têm integrado o quadro de técnicas aplicadas à resolução do problema de caminho mínimo em grafos. Uma área que tem produzido importantes trabalhos é a que envolve roteamento de redes de computadores, cuja função é minimizar a rota dos pacotes de dados transmitidos. O roteamento é um dos processos mais importantes que possuem impacto significativo sobre o desempenho da rede, especialmente pela característica dinâmica deste ambiente.

Um algoritmo de roteamento ideal deve se esforçar para encontrar um melhor caminho para a transmissão de pacotes dentro de um tempo especificado, de modo a satisfazer a Qualidade de Serviço *(QoS)* [29].

Algoritmos Genéticos são capazes de encontrar múltiplas soluções em uma única simulação executada, devido à sua abordagem baseada em população. Desta forma, conseguem se adaptar a situações inesperadas, sejam estas aleatórias ou deliberadas, utilizando-se deste conjunto de soluções durante o processo dinâmico de roteamento [29].

O processo de funcionamento de um Algoritmo Genético começa com a criação de uma população de indivíduos. Em geral, existem dois problemas que devem ser considerados para a inicialização da população: o seu tamanho inicial e sua inicialização.

O tamanho da população é definido com base no problema a ser solucionado. Se este tamanho é demasiado pequeno, não é provável que o algoritmo encontre soluções de grande qualidade. No entanto, se o tamanho da população é muito grande, o algoritmo irá desperdiçar o tempo de processamento, provocando a demora de sua convergência [4].

Outra questão referente à população refere-se às duas maneiras de inicializá-la: de forma heurística ou aleatória. Embora a aptidão média da inicialização heurística seja alta, o que ajuda o algoritmo a encontrar soluções mais rápidas, ela provoca que o algoritmo explore apenas uma pequena parte do espaço de solução [4]. Consequentemente, ele pode nunca se aproximar da solução ideal, devido à falta de diversidade na população. Em uma inicialização aleatória esse risco torna-se remoto.

Cada indivíduo, ou *cromossomo*, pertencente a essa população, representa uma solução codificada candidata para o problema. Apesar de existirem várias formas de representação dos cromossomos, nos problemas com caminhos mínimos em grafos, normalmente adota-se um método de codificação direta, cujos *genes* são representados pelos números inteiros que identificam os vértices. Sendo assim, cada cromossomo corresponde um caminho a partir do vértice de origem até o vértice de destino.

Cromossomos são compostos por genes em sequência. Cada gene identifica um vértice específico, e o seu respectivo *locus* representa a posição em que ele está localizado

no cromossomo. O comprimento dos cromossomos para o contexto dos caminhos mínimos costuma ser variável, pois caminhos de tamanhos diferentes podem ser encontrados, desde que este comprimento não exceda o número total de vértices do grafo.

Dessa forma, um cromossomo compreende uma lista de nós ao longo do caminho construído ($S \rightarrow N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow N_{k-1} \rightarrow N_k \rightarrow D$), como ilustrado na Figura 3.9 [4].

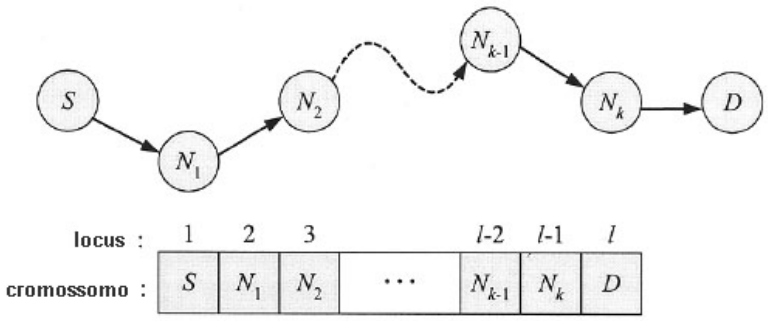


Figura 3.9: *Representação de um Cromossomo.*

Existem diversas maneiras de criar cada cromossomo da população utilizada pelo Algoritmo Genético. A maneira que tem se mostrado mais eficiente em minimizar custos computacionais utiliza a lista de adjacência de cada vértice. Por conseguinte, os genes são escolhidos aleatoriamente de forma a satisfazer as restrições dos vértices de vizinhança e de factibilidade dos cromossomos simultaneamente.

Entende-se por factibilidade nesse contexto, vértices intermediários desconsiderando origem e destino, que ainda não foram escolhidos para integrar o cromossomo, o que evita a formação de *loops*. O processo inicia no nó origem e termina no nó destino, em um processo similar ao da busca em profundidade em grafos [55].

Finalizado esse processo de geração dos indivíduos iniciais, os mesmos são submetidos à avaliações de mérito individual. A intenção é gerar oportunidades reprodutivas de forma que os cromossomos que representam as melhores soluções tenham mais chances de se reproduzir. A definição de solução melhor ou pior está relacionada ao problema e é quantificada de acordo com uma função de aptidão.

Para o problema de caminhos mínimos, a função de aptidão analisa a distância entre cada par de nós em sequência no caminho representado por um cromossomo. Este valor totalizado refere-se ao custo associado ao caminho, que pela relação com o problema, deve ser minimizado. A função de aptidão fornece uma medida da proximidade da solução em relação ao parâmetro de distância, visando encontrar o ponto ótimo.

Uma vez avaliados os cromossomos, diversos operadores genéticos inspirados na genética de Mendel, entram em cena com o intuito de produzir indivíduos mais aptos. Uma característica dos Algoritmos Genéticos é a seleção de indivíduos para a próxima geração. O processo de seleção consiste na escolha dos melhores indivíduos que

funcionarão como cromossomos reprodutores. Desta forma, aqueles com maior aptidão são selecionados para reprodução, enquanto que os outros são descartados.

Existem diversas técnicas para a realização da seleção desses indivíduos como: roleta, torneio, elitista, estocástica, dentre outras. De uma forma geral, o operador de seleção destina-se a melhorar a qualidade média da população, dando a cromossomos de alta qualidade uma chance maior de serem transmitidos para a próxima geração [4].

Através de um processo de recombinação e mutação, a população evolui no sentido de uma solução ótima. Depois de terem sido selecionados para reprodução, os cromossomos são recombinados, utilizando um operador de cruzamento, e mutados, utilizando um operador de mutação, para gerar descendência [12]. Os descendentes deste processo juntam-se aos antigos cromossomos mais aptos para formar uma nova população, a partir do qual o processo é repetido até a convergência.

A primeira operação genética realizada nos cromossomos durante o acasalamento é o cruzamento, ou *crossover*. Com uma probabilidade próxima a 80% de ocorrer, a ideia da operação é criar um intercâmbio de informações entre dois cromossomos, fazendo o algoritmo explorar novos caminhos na tentativa de encontrar soluções melhores.

Existem diversas técnicas para realização do cruzamento genético entre cromossomos tais como: cruzamento com um ponto de corte ou vários pontos de corte, *Partially-Matched Crossover (PMX)*, *Node Based Crossover (NBX)*, etc [4].

Para o problema do caminho mínimo em grafos geralmente é utilizado o operador *NBX*, devido a sua característica particular. Dois cromossomos podem ser cruzados se tiverem pelo menos um gene em comum, desconsiderando-se os genes correspondentes aos vértices de origem e de destino. Se houver mais pares de genes comuns, um par é escolhido aleatoriamente e os locus de cada nó tornam-se um local de passagem dos cromossomos. Nessa técnica, os locus de ambos os indivíduos não precisam ser os mesmos.

O cientista da computação Chang Ahn apresentou uma exemplificação dessa técnica, detalhando a situação dos cromossomos antes e depois do cruzamento, como pode ser visualizado na Figura 3.10 [4].

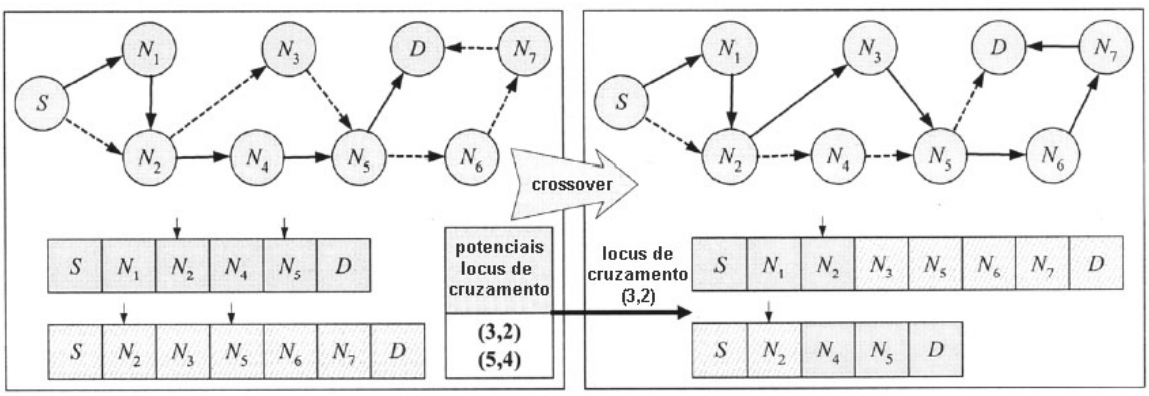


Figura 3.10: *Node Based Crossover (NBX).*

Após realizado o cruzamento com a técnica *NBX*, há uma possibilidade de serem criadas rotas com *loops*. A fim de evitar o problema, uma função reparadora é usada como medida preventiva. *Loops* podem ser reparados através da realização de uma pesquisa ao longo do cromossomo para detectar nós repetidos [4]. Os nós entre o maior *loop* são então removidos, como exemplificado por Ahn na Figura 3.11 [4].

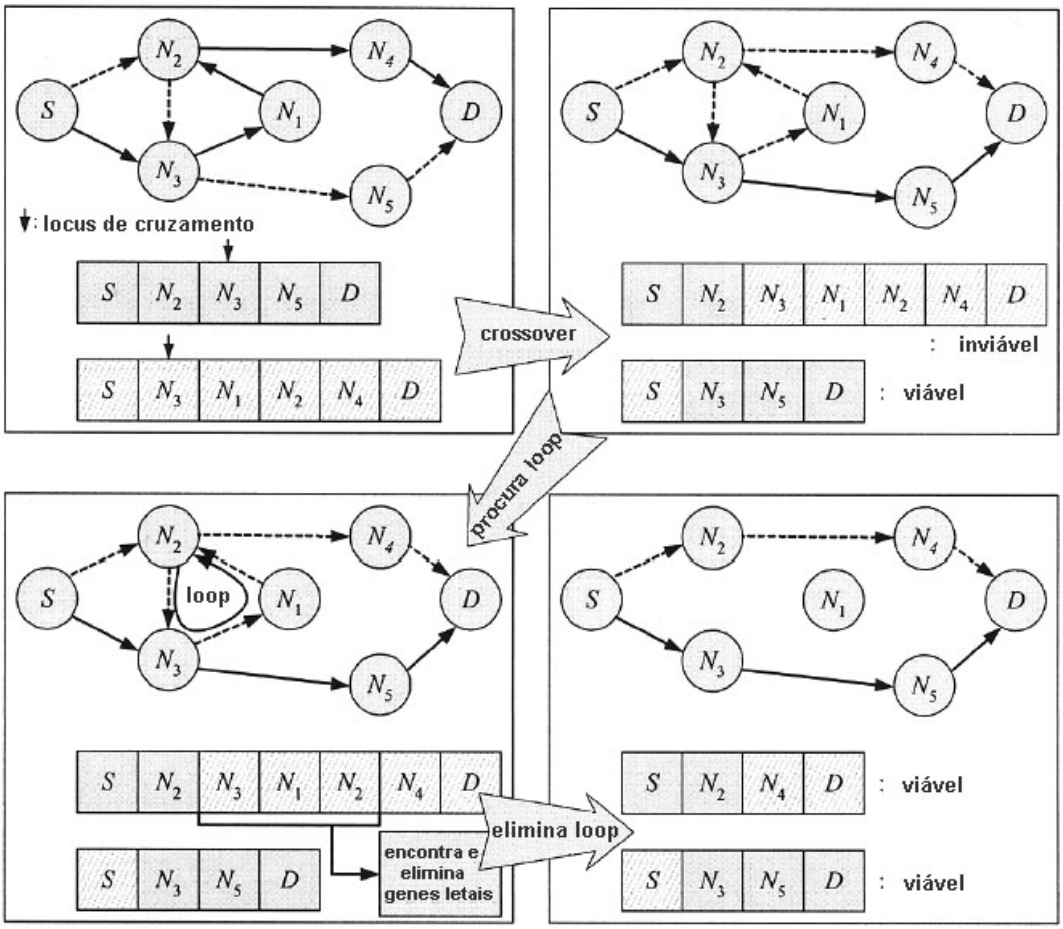


Figura 3.11: *Processo de Reparação do Cromossomo.*

Para evitar que o Algoritmo Genético convirja muito cedo para mínimos locais, a operação de mutação é realizada com pequena probabilidade. Esta operação promove a alteração de um ou mais genes de um cromossomo sorteado aleatoriamente.

Assim como a operação de cruzamento, a mutação também pode ser realizada de diversas maneiras, utilizando-se de técnicas como intercâmbio, inversão, inserção, remoção, dentre outras. Os pontos de mutação são selecionados aleatoriamente no cromossomo, e o nó mutado deverá também respeitar as restrições de adjacência dos genes e de factibilidade do cromossomo. Com a conclusão da mutação, a descendência gerada tem de ser validada com o mesmo processo usado no cruzamento. Um exemplo da operação de mutação sendo realizada é apresentado por Ahn na Figura 3.12 [4] .

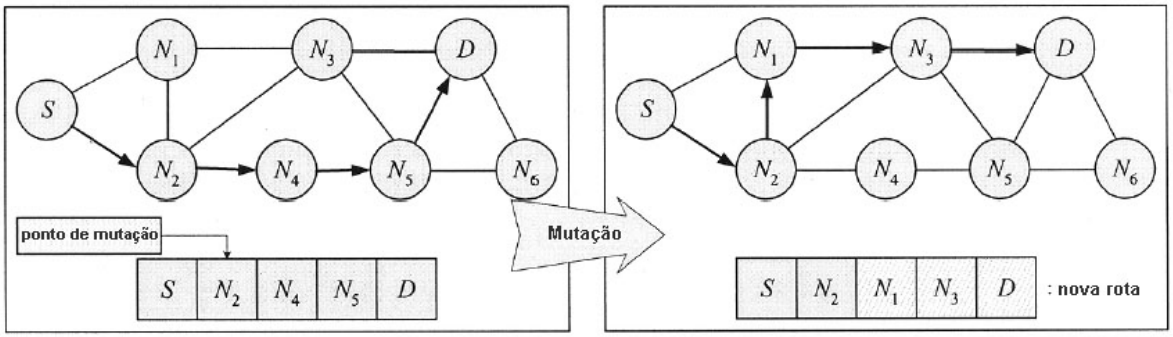


Figura 3.12: *Processo de Mutação do Cromossomo.*

Finalmente, a condição de terminação do Algoritmo Genético pode ser estabelecida por um número fixo de iterações, ou quando nenhuma mudança na aptidão da população é verificada durante certo período de tempo.

O limite superior para o tempo de otimização do Algoritmo Genético, na busca de caminhos mínimos em grafos ponderados, foi determinado por Scharnow como $O(n^2 l \log n)$, onde n é o número de vértices e l, o número máximo de arestas em um suposto caminho mínimo que conecta o vértice de origem ao vértice de destino [89].

O algoritmo 3.3 apresenta o *Algorimo Genético* em sua forma padrão.

Algoritmo 3.3: $GA(G, w, s, t)$

Entrada: grafo $G(V, E)$, vetor de pesos não negativos das arestas w e vértices de origem s e destino t.

Saída: vetor de vértices P^*, representando o cromossomo mais adaptado.

```
1 criar aleatoriamente uma população P de indivíduos
2 determinar a qualidade de P
3 enquanto condição de parada não satisfeita faça
4     selecionar um subconjunto N de P para reprodução, de acordo com sua
      qualidade
5     usar, estocasticamente, o operador de cruzamento sobre cada par de N,
      gerando dois (novos) filhos
6     usar, estocasticamente, o operador de mutação sobre cada indivíduo
      resultante da etapa anterior
7     atualizar P selecionando indivíduos de P ∪ N
8     avaliar qualidade dos indivíduos da nova população P
9 fim
```

CAPÍTULO 4

Agentes e Sistemas Multiagentes

Independente da classificação que os seres humanos adotam para os efeitos advindos da complexidade, o que parece certo é a onipresença dos mesmos na natureza, assim como brilhantemente exemplificou Duncan Watts [106]:

> Se quisermos desencadear uma avalanche nas montanhas, podemos lançar uma bomba atômica, mas isso não seria necessário. Um único esquiador passando pelo tipo errado de neve, na parte errada da montanha, na hora errada do dia, pode liberar uma fúria totalmente desproporcional à pequenez da origem.

Condicionados por essa excêntrica conduta da natureza, os últimos anos da existência humana foram invadidos pela emergência artificial. Sistemas construídos com o conhecimento consciente do que é a emergência, capazes de explorar suas leis para a resolução de problemas reais, muitos deles verdadeiramente complexos.

O que une esses diferentes sistemas é um padrão recorrente: uma rede de auto-organização, de agentes dessemelhantes que, embora muitas vezes constituídos por elementos simplórios, inadvertidamente criam uma ordem de nível superior [60].

O termo *agente* domina a Inteligência Artificial e invadiu definitivamente a própria informática. Uma visão centralizadora determinou no início desta ciência, a concepção do agente só e isolado, dependente do seu interlocutor humano.

Tal concepção só sofreu modificações quando surgiu o movimento da descentralização da computação, no fim dos anos 70. A consagração da distribuição, ao longo da década de 80, possibilitou o diálogo com as Ciências Sociais, e apoiou posteriormente o regresso das Neurociências à trama multidisciplinar da sua base de apoio [90].

A hierarquização do poder, outrora em voga, foi substituída por novas formas de organização, alicerçadas em experiências plurais. Em vez de comportamentos isolados, surgiram comportamentos coletivos, através da coordenação e da cooperação entre agentes em situações de grande complexidade [90]. Foi a partir deste momento que se conduziram as novas incursões em prol de uma renovação dos métodos e das técnicas, garantindo condições para a entrada segura na segunda era da Inteligência Artificial.

O que se segue é um circuito por áreas que não costumam estar incluídas em um mesmo trabalho como: os êxitos do *Deep Blue II* sobre Gary Kasparov; o poder

de mecanismos de raciocínio probabilístico automático para enfrentar a incerteza; a exploração de técnicas de procura para as máquinas de busca na Internet; o uso do *software* de animação *Massive*, para as cenas de batalha da trilogia *O Senhor dos Anéis*; o programa de composição musical *SARA*, capaz de imitar Bach, Chopin, Mozart [90].

Atualmente a Inteligência Artificial busca soluções para a construção de agentes autônomos capazes de aprender e de lidar com a incerteza, além de serem capazes de evoluir, adaptando-se aos múltiplos ambientes em que podem viver. Todavia, tais agentes não percorrem este sinuoso caminho de forma isolada, mas através de sociedades de agentes que cooperam na resolução de problemas verdadeiramente complexos.

4.1 Agentes

Apesar de inexistir uma definição singular, entende-se por agente toda entidade capaz de interagir com o ambiente, guiado em geral, mas não unicamente, por objetivos [90]. Estruturalmente, um agente é algo simples, como ilustrado na Figura 4.1 [90].

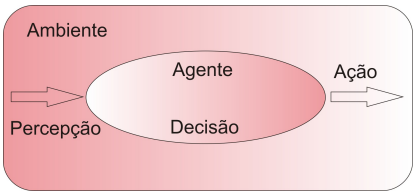


Figura 4.1: *Arquitetura de um Agente.*

Basicamente, um agente possui mecanismos que lhe permite recolher informação do ambiente *(percepção)*, bem como atuar sobre o mesmo *(ação)*. Além disso, um agente possui processos que viabilizam sua definição sobre a melhor ação a ser realizada *(decisão)*, sendo tais processos tão complexos quanto sua tarefa ou seu ambiente [90].

Alicerçado em suas propriedades, um agente cria comportamentos estratégicos que lhe permitem ser casualmente bem sucedido em suas tarefas, sejam elas impostas pelo ambiente, ou determinadas por seus objetivos. A Figura 4.2 [90] ilustra esta perspectiva.

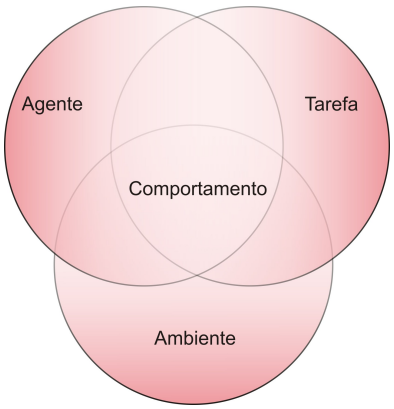


Figura 4.2: *Perspectiva Funcional de um Agente.*

Os agentes podem apresentar muitas variações de comportamento, o que está relacionado com o subconjunto de propriedades pertinente a cada um. Algumas das mais importantes propriedades idealizadas para um agente são [12]:

- **Autonomia:** capacidade de adotar ações direcionadas à execução das tarefas e dos objetivos a serem alcançados, sem a interferência do usuário final;
- **Aprendizagem:** permite a alteração de seu comportamento baseado em experiências anteriores;
- **Comunicabilidade:** capacidade para acessar informações sobre o estado atual do ambiente;
- **Confiabilidade:** credibilidade apresentada na realização de suas ações e a certeza de que não processará informações falsas;
- **Cooperatividade:** capacidade para trabalhar junto com outros agentes, objetivando a realização de uma tarefa em comum;
- **Degradação:** capacidade para completar a execução de uma tarefa, mesmo quando ocorre alguma anomalia no sistema;
- **Inteligência:** capacidade de negociar em situações novas de incerteza;
- **Mobilidade:** capacidade de transportar-se de um local para outro;
- **Persistência:** habilidade de manter-se preciso através do tempo;
- **Personalidade:** capacidade para demonstrar opinião.

Com base na ideia de um conjunto de propriedades é possível definir uma hierarquia de agentes, através da qual os mesmos dividem-se inicialmente em três grandes grupos: biológicos, robóticos e computacionais. A Figura 4.3 [90] ilustra esta hierarquia.

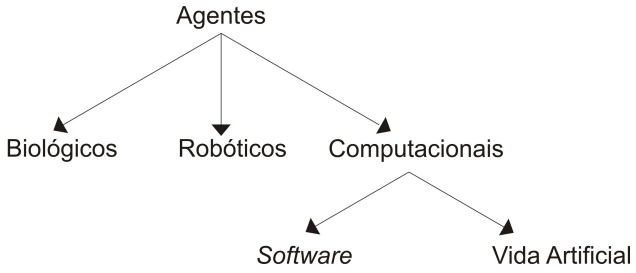


Figura 4.3: *Hierarquia de Agentes.*

Os primeiros caracterizam os agentes naturais. Os segundos são os agentes artificiais em que o corpo assume um papel fundamental. Finalmente, os terceiros somente

existem como programas de computador, sendo subclassificados em dois grupos distintos: os agentes de *software* e os agentes de vida artificial. Enquanto os primeiros são supostos serem eternos, os segundos morrem [90]. Normalmente, é pensando nos agentes de *software* que se usa a palavra agente.

As tarefas de um agente geralmente são variadas, o que implica em classificações por diferentes perspectivas. Segundo Ernesto Costa, toda tipologia de tarefas possui subjetividades, porém os seguintes aspectos permitem que elas sejam caracterizadas [90]:

- **Deliberação:** algumas tarefas obrigam o agente a possuir processos de decisão sofisticados enquanto outras não;
- **Mudança:** algumas tarefas envolvem mudanças importantes, seja ao nível do agente, como mecanismos de decisão, seja ao nível do próprio ambiente;
- **Interação:** muitas tarefas só podem ser resolvidas graças a uma forte interação entre o agente e o ambiente;
- **Imposição:** existem tarefas que são determinadas essencialmente pelo ambiente.

Os agentes têm de enfrentar diferentes tipos de ambientes, que variam em graus de complexidade. Além disso, o modo como um agente analisa e compreende o ambiente depende fortemente de suas capacidades. Na visão de Costa existem três aspectos principais que podem ser usados para classificar os ambientes [90]:

- **Acessíveis:** o agente pode retirar do ambiente toda a informação que necessita para determinar a melhor ação;
- **Deterministas:** quando a evolução do ambiente pode ser determinada de forma única a partir da situação corrente e da ação do agente sobre o ambiente;
- **Estáticos:** não se alteram, enquanto o agente está decidindo a ação a ser executada.

Os ambientes mais complexos são aqueles que são inacessíveis, não deterministas e dinâmicos, conforme visualizado no ponto 2 da Figura 4.4 [90]. No outro extremo, os ambientes caracterizam-se por serem acessíveis, deterministas e estáticos, como ilustrado no ponto 1 da mesma figura, onde residem as situações mais favoráveis para o agente.

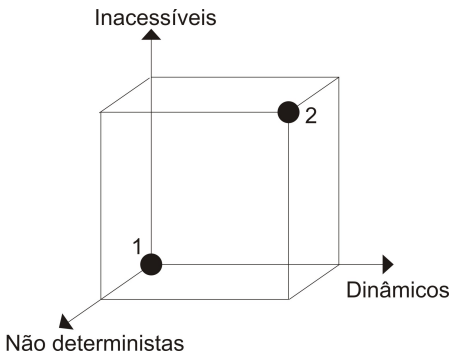


Figura 4.4: *Tipos de Ambientes.*

O século XXI habilitou os seres humanos a lidar com agentes que executam tarefas específicas. Agentes de interface, que tentam fazer o que os usuários pretendem e não o que informam quando interagem com uma aplicação. Agentes de procura, que na Web tentam encontrar informação relevante. Agentes de filtragem, que permitem aos correios eletrônicos selecionar mensagens e arrumá-las devidamente. Agentes assistentes, específicos de um dado domínio, que ajudam as pessoas a manter suas agendas organizadas.

Em algumas dessas situações, especialmente as caracterizadas por grande complexidade, os agentes vivem mergulhados em ambientes dotados de outros agentes, que interagem entre si de formas díspares durante a resolução de suas tarefas, formando em conjunto um Sistema Multiagente.

4.2 Sistemas Multiagentes

Uma população de agentes, cada um possivelmente com estratégias particulares, que se unem em prol da criação de um macrocomportamento observável, caracteriza um Sistema Multiagente. Em diversas situações reais é interessante utilizar as habilidades de diferentes agentes para resolver problemas, e os Sistemas Multiagentes são exímios combinadores de tais habilidades. Além disso, apresentam maior rapidez na resolução de tarefas complexas, devido ao paralelismo que pode ser obtido [12]. É perceptível também nestes sistemas uma melhora na confiança dos resultados, pois podem ser incluídos agentes capazes de validar e corrigir os resultados outrora fornecidos por seus pares.

Por fim, a modularidade obtida com os Sistemas Multiagentes é outra característica relevante. Caso o sistema não consiga resolver uma tarefa específica, a utilização de um novo agente, melhor projetado ao contexto, pode ser realizada, sem a necessidade da completa substituição do sistema.

A arquitetura mais simples e utilizada no controle desses agentes é a arquitetura *Quadro Negro*. Esta arquitetura não prevê uma comunicação direta entre os agentes, sendo todas as comunicações feitas por uma estrutura central compartilhada [12].

A estratégia oposta ao *Quadro Negro* propõe a troca de mensagens entre os próprios agentes, o que os obriga a possuir uma identificação precisa, como um nome único no sistema e, também, a adoção de algum protocolo de comunicação [12].

Outra diferenciação importante é quanto ao tipo de interação realizada pelos agentes no sistema. Um grupo de agentes que cooperam na resolução de problemas que estão além da capacidade de resolução individual de um agente, caracteriza um sistema colaborativo. Não obstante, quando os agentes não colaboram por um objetivo em comum, trabalhando apenas por uma meta particular, o sistema é dito como concorrente [6].

Quanto à composição de seus elementos individuais, os Sistemas Multiagentes podem ser classificados como homogêneos ou heterogêneos. O primeiro caracteriza um

grupo de agentes idênticos, possuidores dos mesmos objetivos, ações e domínios de conhecimento. Já o sistema heterogêneo é constituído por agentes distintos, que de alguma forma possuem objetivos, ações ou domínios de conhecimento diferentes [6].

A organização dos Sistemas Multiagentes pode ser classificada de diferentes maneiras, sendo as mais comuns [12]:

- **Hierárquica:** o controle e a tomada de decisão são concentrados em níveis, ou seja, a interação entre os agentes ocorre através de uma comunicação vertical do agente superior para os agentes subordinados;
- **Comunidade de especialistas:** a organização mantém todos no mesmo nível, porém cada agente tem uma especialidade em certo domínio. Nesta organização, a interação entre os agentes ocorre de acordo com regras previamente estabelecidas;
- **Comunidade científica:** as soluções para os problemas são localmente construídas e, em seguida, testadas e refinadas, a partir de uma comunicação com outros agentes solucionadores de problemas.

De maneira geral, Sistemas Multiagentes utilizam-se de agentes interativos que criam um comportamento de nível superior em uma escala acima da ocupada pelos agentes individuais, e por isso adaptam-se apropriadamente para diversos tipos de ambientes. As aplicações fazem-se cada vez mais presentes na história da humanidade, atuando em áreas tão díspares quanto a Web, os Jogos ou a Medicina.

Não é por acaso que as diversas escolas da Inteligência Artificial, a simbólica, a conexionista, a evolucionária e a robótica, transformaram a perspectiva reducionista, em uma perspectiva holística e global. Desta forma, aceitaram com humildade que cada uma das escolas apenas contempla uma parcela sobre os fenômenos da inteligência, e que estas parcelas são complementares mantendo relações entre si.

Abaixo, ilustrado na Figura 4.5 [2], um dos maiores e mais misteriosos exemplos de sistema complexo adaptativo, e que indubitavelmente é objeto de pesquisas e inspirações para os Sistemas Multiagentes da Inteligência Artificial.

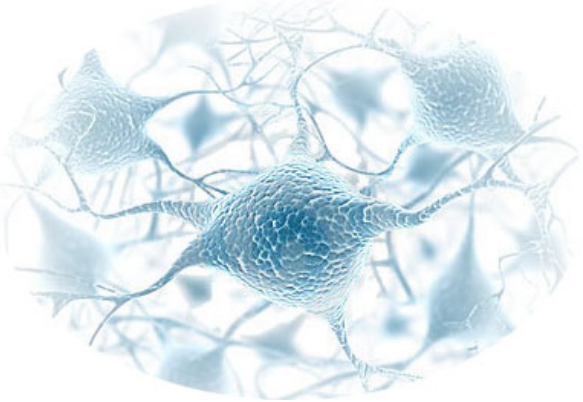


Figura 4.5: *Neurônios do Cérebro Humano.*

CAPÍTULO 5

Visualização da Informação

Representações gráficas de toda sorte têm sido usadas como instrumento de comunicação desde os primórdios da humanidade. Há milênios, no Egito, seus habitantes utilizavam uma linguagem baseada em símbolos, os hieróglifos [37]. Este modelo de linguagem era composto por representações gráficas, figuras que expressavam alguma informação, transmitindo mensagens que inclusive atualmente podem ser compreendidas.

O oferecimento de imagens, ou qualquer outro tipo de recurso gráfico com o intuito de apresentar uma informação, na maioria dos casos produz a compreensão da mensagem transmitida, já que esta se torna mais natural aos seres humanos.

No processo cognitivo do ser humano apresentado por Card, as informações captadas pelos órgãos são armazenadas na memória de curta duração (MCD). No caso de uso frequente de uma informação, sua transferência é realizada para a memória de longa duração (MLD). A MCD é formada por *chunks*, elementos ativados da MLD que podem ser organizados em unidades maiores. O *chunk* é função tanto do usuário quanto da tarefa a ser realizada, uma vez que se trata da ativação de sua MLD [27].

Para esclarecer esse processo, Dias apresenta o seguinte teste. Se a sequência das letras V-I-B-R-M-C-D for lida sem qualquer entonação ou separação de tempo, pode ser difícil lembrá-la posteriormente. Todavia, quando lida com entonação e separação: Visualização da Informação – BR – MCD, o processo de memorização é facilitado. No primeiro caso 7 *chunks* devem ser recordados, enquanto no segundo, apenas 3 [37].

Os sentidos são a base da percepção humana, que se caracteriza por um sistema sensorial constantemente estimulado por um fluxo contínuo de acontecimentos envolventes. Apesar das sensações externas valerem-se dos cinco sistemas sensoriais, recursos gráficos exploram essencialmente o sentido humano mais apto a captar informação temporal: a visão. Além de ser o primeiro componente do sistema sensorial, a visão é o sentido adquirido mais rapidamente pelo cérebro [8].

Outra característica importante desse sentido é sua capacidade de paralelismo, o que lhe permite observar a circunvizinhança de um determinado objeto visual, mesmo tendo a atenção focada em um ponto específico do mesmo [8]. Esta capacidade permite aos seres humanos o reconhecimento de padrões, como ilustrado na Figura 5.1 [75].

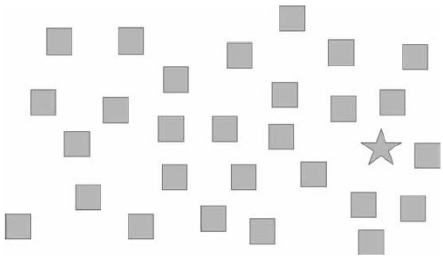


Figura 5.1: *Detectando Objeto Dessemelhante.*

Em seus experimentos, Dias e Carvalho inspirados em Card, apresentam a capacidade da visão captar informações através de imagens em comparação com palavras. A Figura 5.2 mostra uma adaptação de um destes experimentos, que consistia no processo de detecção de triângulos inseridos em um ambiente repleto de quadrados [37].

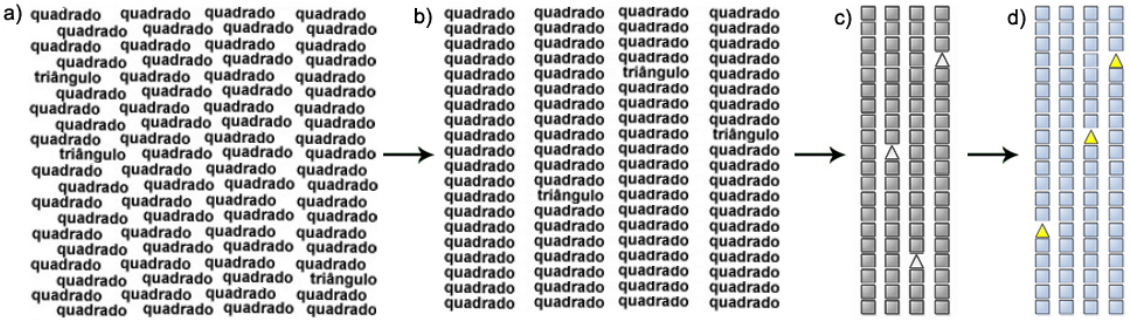


Figura 5.2: *Melhoria da Percepção Humana.*

Durante um tempo t, constata-se que existem três palavras triângulo na situação (a), sendo as mesmas apresentadas de forma desalinhada. Alterando a configuração do experimento, de acordo com a situação (b), é possível obter um melhor desempenho na busca. Isto ocorre devido ao alinhamento das palavras, que permite à cognição humana a possibilidade de detectar e comparar padrões [37].

Ao repetir a mesma experiência, agora aplicada na situação (c), é possível notar com facilidade, a presença dos triângulos através do uso de símbolos. Finalmente, na situação (d), observa-se a utilização de outro importante recurso da representação gráfica, a saturação de cor, que torna a tarefa do experimento trivial. Esta exemplificação demonstra como a representação de informações por intermédio de recursos gráficos pode ser poderosa, contribuindo com o processo cognitivo humano.

Um importante acontecimento da história da ciência foi envolvido diretamente pelo forte poder da percepção visual humana, provando que muitos dados abstratos podem ser condensados em uma simples imagem.

A cidade prussiana de Königsberg é banhada pelo rio Pregel que, ao cortar a cidade se ramifica formando uma ilha, que no século XVIII se ligava ao restante da cidade por sete pontes, como exemplificado à esquerda da Figura 5.3. Os habitantes da cidade

tentavam efetuar um percurso que atravessasse todas as pontes, porém passando somente uma única vez em cada. Como eles sempre falhavam em suas tentativas, acabavam acreditando que não era possível encontrar tal percurso, sem saberem o real motivo [10].

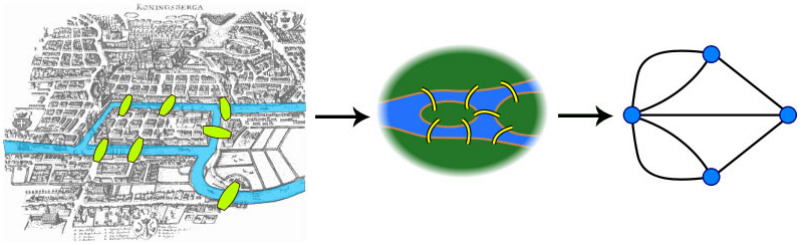


Figura 5.3: *Pontes de Königsberg.*

Em 1736, Leonard Euler, um dos maiores matemáticos de todos os tempos, conseguiu decifrar tal enigma, se valendo de um recurso fundamental. Usando um raciocínio simples, o matemático transformou os caminhos em retas e suas intersecções em pontos, criando possivelmente o primeiro grafo da história [14]. A sequência do raciocínio do matemático pode ser acompanhada no centro e à direita da Figura 5.3.

Euler percebeu que só seria possível atravessar o caminho sob as restrições impostas, se houvesse exatamente zero ou dois pontos de onde saísse um número ímpar de caminhos, restrição não satisfeita pelas condições estruturais da cidade [14]. Neste ponto, Euler não só resolveu o problema de Königsberg, mas acabou desbravando um imenso ramo da matemática, a Teoria dos Grafos, muito particularmente devido à sua forte percepção ao reconhecer padrões na imagem do grafo gerada.

Um modelo simplificado do sistema de processamento de informação através da percepção visual humana, invariavelmente se torna fundamental para análises mais aprofundadas neste contexto. Tal sistema foi dividido por Colin Ware em três fases [104]:

- **Processamento paralelo para extrair propriedades de baixo nível da cena visual em causa:** A informação visual é a primeira a ser processada por bilhões de neurônios que paralelamente extraem características de partes da imagem visual considerada. Tais neurônios recuperam informações como orientação dos contornos, cor e textura, determinando o principal ponto visual da imagem analisada;

- **Percepção de padrões na imagem formada:** Na segunda etapa, processos ativos decompõem rapidamente o campo visual em regiões e padrões simples, como contornos contínuos e regiões de cores ou textura semelhantes. Os padrões de movimento são extremamente importantes nesta fase;

- **Processamento sequencial dirigido:** Num nível superior da percepção estão as imagens contidas na memória visual através das demandas da atenção ativa, e

será esta memória que contribuirá com as pesquisas visuais. É neste nível que a informação armazenada anteriormente permitirá a construção de padrões.

Em resumo, a utilização de técnicas de visualização de informações para ampliar a cognição sobre dados abstratos tem um forte apelo quando comparada com outras formas de transmitir ou analisar informações. Foi esta percepção que possibilitou o desenvolvimento de uma nova área repleta de desafios, a Visualização da Informação.

Proposta inicialmente por Robertson, Card e Mackinlay em 1989, a Visualização da Informação tem por objetivo o estudo das principais formas de representações gráficas de informações, a fim de contribuir para o seu entendimento, bem como favorecer a percepção de novos conhecimentos [37].

Os estudos sobre o assunto recebem contínuas contribuições advindas de diversos ramos da ciência. Sendo assim, a Visualização da Informação se encontra primeiramente relacionada à Psicologia, à Linguística e às Artes Visuais, no que se refere à forma como o ser humano vê e interpreta imagens. Não obstante, ela também possui grande relação com subáreas da Computação, como Computação Gráfica, Visão Computacional, Interação Homem-Computador e Mineração de Dados [75].

A Visualização da Informação tem se beneficiado com as inovações geradas pela evolução dos computadores, os quais proporcionam meios cada vez mais eficientes de melhorar a geração de imagens e de aumentar a interatividade em tempo real. Assim, ferramentas computacionais de visualização podem apoiar os indivíduos no processo de análise dos dados, favorecendo três atividades [8]:

- **Análise exploratória:** Descobrimento de novos conhecimentos contidos nos dados, através de um processo analítico de exploração de sua representação visual, com subsequente procura de indícios que possam indicar tendências ou relações;

- **Análise confirmatória:** Determinação de evidências para aceitação ou rejeição de hipóteses específicas;

- **Apresentação:** Representação gráfica e apresentação do relacionamento, estrutura, comportamento e outras características intrínsecas aos dados analisados.

É de fundamental importância perceber o recente papel da Visualização da Informação, que incluiu a ferramenta computacional como suporte natural ao processo de apresentar e interagir com visualizações. Tal cenário contrasta-se com as visualizações da época pré-computacional, formadas basicamente por imagens estáticas, desprovidas de qualquer interatividade.

A Figura 5.4 [75] ilustra um modelo de referência para o processo de visualização de informações apresentado por Card [26], constituído por três etapas.

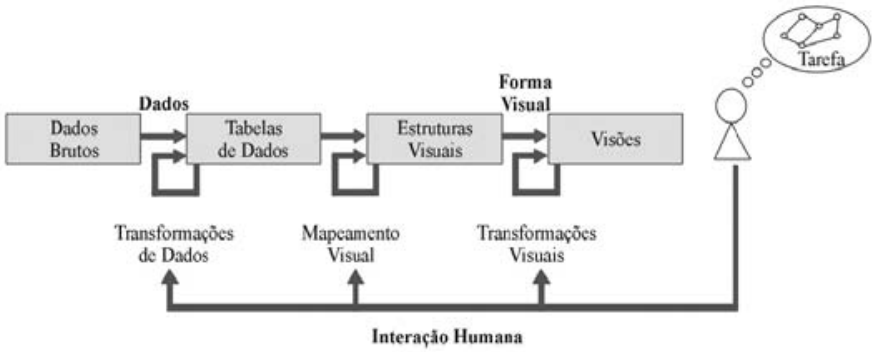
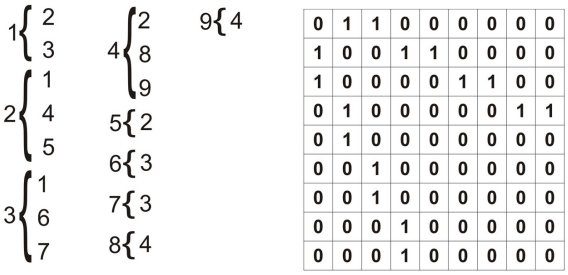


Figura 5.4: *Modelo de Referência.*

A primeira etapa é chamada de *Transformações de Dados*, na qual um conjunto de dados brutos é organizado em uma forma lógica mais estruturada, geralmente na forma de tabelas. A Figura 5.5 exemplifica este processo, mapeando em uma matriz de adjacência as relações entre os elementos do modelo à esquerda da figura.



0	1	1	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0
1	0	0	0	0	1	1	0	0
0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0

Figura 5.5: *Transformações de Dados.*

A próxima etapa contempla o *Mapeamento Visual*, ou a construção de uma composição gráfica que visualmente represente os dados. Este mapeamento se decompõe em três partes: substrato espacial, marcas visuais e propriedades gráficas das marcas.

O substrato visual caracteriza o espaço para a visualização, geralmente representado por eixos, como os eixos X e Y do plano cartesiano. As marcas visuais são símbolos gráficos utilizados para representar os itens de dados. A Figura 5.6 [75] demonstra os símbolos mais comuns como pontos, linhas, áreas, volumes e ícones.

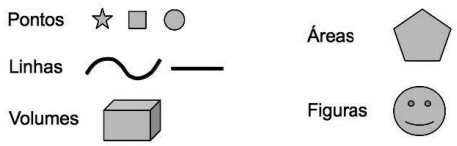


Figura 5.6: *Símbolos Gráficos Representacionais.*

As propriedades gráficas são os atributos visíveis que caracterizam as marcas visuais. Algumas propriedades amplamente utilizadas são: a posição da marca dentro do substrato espacial, a forma, a cor, o tamanho, a área ou o volume, a orientação, o sentido e

a inclinação do tipo de linha, a textura, dentre outras. A Figura 5.7 realiza o mapeamento visual dos dados apresentados na Figura 5.5.

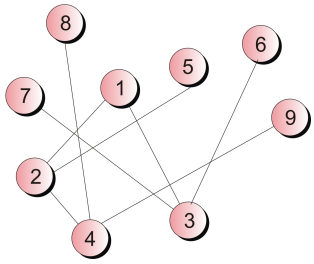


Figura 5.7: *Mapeamento Visual.*

As *Transformações Visuais* são a última fase desse processo, no qual é possível modificar e estender as estruturas visuais interativamente através de operações como:

- **Testes de localização:** possibilitam obter informações adicionais sobre um item da tabela de dados;
- **Controles de ponto de vista:** permitem ampliar, reduzir e deslocar a imagem com o objetivo de oferecer visões diferentes;
- **Distorções da imagem:** visam criar ampliações de uma região específica em detrimento de outra.

Segundo Nascimento, os mecanismos de interação implementados nessa etapa possibilitam ao usuário explorar diferentes cenários, ou perspectivas, para uma melhor compreensão dos dados visualizados. O autor também enfoca que o esforço de exploração destes dados é repassado em parte para o computador, que redesenha a imagem enquanto o usuário observa como a visualização se modifica [75].

Muitas vezes padrões podem ser facilmente descobertos pelo ser humano nesse processo, como exemplificado na Figura 5.8, onde emerge uma estrutura hierárquica em uma diferente visualização do grafo da Figura 5.7.

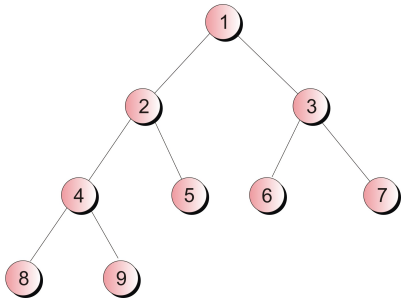


Figura 5.8: *Transformações Visuais.*

É importante ressaltar que nem toda visualização de informação é útil para o usuário, sendo dois aspectos cruciais para este fim: a expressividade e a efetividade [71].

Uma visualização pode ser considerada expressiva quando é capaz de apresentar todos os dados de interesse do usuário, sem qualquer informação sobressalente. Já a efetividade se relaciona com a apresentação das informações de forma clara, objetivando sua percepção de forma fácil e rápida, minimizando possíveis erros de interpretação.

De acordo com Nascimento, expressividade e efetividade são aspectos essenciais, pois suas ausências implicam em visualizações incapazes de enfatizar padrões relevantes nos dados, não acrescentando algo significativo ao já trivialmente conhecido [75].

Mesmo em uma época em que Computação e Visualização da Informação eram estudos distantes, o grafo construído por Euler representou as pontes de Königsberg de formas expressiva e efetiva, contribuindo fundamentalmente para o avanço da ciência.

5.1 Técnicas de Visualização de Informações

A escolha da metáfora visual mais adequada para visualização de estruturas é fundamental para facilitar o entendimento humano sobre os dados. Nesse sentido, nas várias técnicas de visualização de informações, frequentemente, os autores buscaram inspiração em objetos do mundo real ou geométricos, para mapear o conjunto de informações.

Dentre as técnicas mais utilizadas destacam-se [75]:

- **Grafos:** Utilizado para demonstrar relações entre objetos ou pessoas ou estruturas hierárquicas;
- **Fisheye:** Permite uma visão mais detalhada de uma região de interesse como o utilizado no menu do *Mac OS*;
- **Browser Hiperbólico:** Destacam uma área específica de interesse com desenho radial de árvores, para auxiliar na exploração de grandes hierarquias;
- **Bifocal Display:** Itens de informação são apresentados em três áreas distintas, uma central que contém a informação em foco, e o restante nas laterais;
- **Perspective Wall:** Utiliza visões detalhadas e contextuais para suportar a visualização linear de espaços de informações estruturadas;
- **Coordenadas paralelas:** Mapeia um espaço n-dimensional em uma estrutura bidimensional que usa n eixos paralelos verticais equidistantes.

Para o propósito desta dissertação, grafos são estruturas visuais interessantes, pois, além de serem intuitivos, possuem alto poder como ferramenta de modelagem e representação de dados relacionais. Muitos dos aplicativos que utilizam grafos necessitam deste poder para que a informação que eles representam possa ser melhor compreendida e utilizada. Assim, a construção de grafos desempenha um papel importante no campo de Visualização da Informação, sendo aplicada em diferentes áreas da ciência.

CAPÍTULO 6

Frameworks Multiagentes Java

A característica essencial das plataformas livres consiste na abstração de conhecimento viabilizada aos seus usuários, que usufruindo de seus importantes recursos, conseguem criar inovações em diferentes escalas.

Uma das plataformas mais produtivas que se desenvolveu notoriamente foi a Web. Segundo Johnson, "a Web pode ser imaginada como uma espécie de sítio arqueológico, com camadas sobre camadas de plataformas enterradas sob cada página" [61].

Em 1990, Tim Berners-Lee foi capaz de projetar sozinho esse poderoso meio porque pôde construir livremente sobre os protocolos abertos da Internet [61]. Logicamente, não foi necessário criar um sistema inteiro para que a comunicação entre computadores se espalhasse mundialmente, um problema outrora resolvido.

Bastou-lhe construir uma estrutura padrão para descrever páginas de hipertexto *(HTML)* e compartilhá-las através de canais da Internet *(HTTP)*. Até mesmo o *HTML* se baseou em outra plataforma, a *SGML*, desenvolvida pela IBM na década de 60 [61].

Menos de duas décadas depois, quando Hurley, Chen e Karim criaram o *YouTube* [48], maior repositório mundial de vídeos, o fizeram combinando elementos de três plataformas distintas: a Web, a plataforma *Flash* [3], e a linguagem de programação *Javascript* [45]. Para Johnson, a habilidade de construir sobre estas plataformas preexistentes explicam como três sujeitos conseguiram criar o *YouTube* em apenas seis meses [61].

Em um processo análogo, o uso de uma plataforma devidamente homologada, e disponibilizada livremente para a criação de sistemas multiagentes torna-se fundamental, pois minimiza o esforço de desenvolvimento de aplicações sob este paradigma. Consequentemente, ao se valer de uma ferramenta com estas propriedades, o desenvolvedor é estimulado a utilizar os recursos que já estão prontos e validados, evitando assim que os mesmos sejam refeitos desnecessariamente.

De acordo com o cientista da computação Jair Alarcón, alguns atributos não podem ser negligenciados em uma plataforma de desenvolvimento para criação de agentes e sistemas multiagentes, destacando-se como essenciais [7]:

- **Linguagem:** utilizar uma linguagem orientada a objetos, ter portabilidade e sintaxe

simples, ser distribuída, aberta, segura, robusta e possuir diversos recursos necessários à programação de sistemas multiagentes;

- **Licença:** poder ser utilizada e modificada gratuitamente sem nenhuma restrição, como especificado pela *FSF (Free Software Foundation)* [94];
- **Atualidade:** possuir algum tempo de surgimento, além de se manter devidamente atualizada, o que indica respectivamente, maturidade e continuidade;
- **Características:** disponibilizar o maior número de serviços e recursos, seguir padrões para implementação de sistemas multiagentes, e ser eficiente;
- **Suporte:** disponibilizar uma documentação adequada através de tutoriais, manuais, guias, livros, além de fóruns e listas de discussão.

Em 2010, Alarcón realizou um breve comparativo entre vinte diversas ferramentas que trabalham sob o paradigma multiagente, objetivando definir a melhor plataforma [7]. *JADE*, *JESS*, *CARTAGO*, *Jadex*, *Zeus*, *AgentBuilder*, *Voyager* e *Grasshopper* são alguns exemplos das ferramentas analisadas de acordo com suas adequações, aos atributos técnicos e de qualidade levantados pelo cientista.

Segundo Alarcón, a escolha pela ferramenta JADE foi feita com facilidade, pois foi a única a atender todos os critérios estipulados em sua avaliação. Valendo-se desta análise pré-concebida pelo cientista, e verificando que a referida plataforma, além de ter sido escrita em Java, mantém-se devidamente atualizada perante a data desta dissertação, a adoção do *framework* multiagente *JADE* foi realizada para compor esta obra.

6.1 JADE - *Java Agent DEvelopment framework*

JADE (*Java Agent DEvelopment framework*) é um *framework*, arcabouço, ou biblioteca de *software*, que possibilita o desenvolvimento de aplicações baseadas em agentes, conforme as especificações da *FIPA (The Foundation for Intelligent Physical Agents)* [59]. A FIPA é uma organização de padronização *IEEE* que promove este tipo de tecnologia, além de viabilizar a interoperabilidade destes padrões com outras abordagens.

Segundo Alarcón, antes das especificações da FIPA existiam cerca de 60 sistemas proprietários de agentes em competição, sendo a maioria fechada e incompatível, refletindo um atraso no desenvolvimento deste tipo de sistema. Atualmente, estas especificações formam o maior movimento de padronização nesta área da tecnologia, sendo o JADE considerado o principal *framework* de código aberto que as implementa [7].

O desenvolvimento do JADE iniciou-se em 1998 pela Telecom Italia. Logo após, no ano de 2000, o *framework* adentrou a comunidade de *software* livre, e desde então, tem

sido distribuído sob a licença *LGPL*, que diferentemente da *GPL*, não impõe qualquer restrição de utilização do *software*. Atualmente, o JADE pode ser adquirido em [96].

Uma característica importante dessa ferramenta se deve ao fato de ser possível distribuí-la através de diversas máquinas com sistemas operacionais diferentes. Outro benefício, refere-se à possibilidade de toda sua configuração, gerenciamento e depuração poderem ser feitos em tempo de execução através de ferramentas gráficas remotas [6].

O JADE possui uma arquitetura de comunicação flexível e eficiente, uma vez que o modelo de comunicação FIPA foi todo implementado. O mecanismo de transporte é capaz de se adaptar a qualquer situação, através da escolha do protocolo mais adequado. Além disso, a maioria dos protocolos de interação FIPA também foi implementada, o mesmo ocorrendo para com os protocolos de gerenciamento de ontologias [6].

6.1.1 Arquitetura do JADE

Na arquitetura do JADE, cada computador possui um contêiner onde vários agentes podem ser executados, como ilustrado na Figura 6.1 [7]. As plataformas podem ser compostas por um conjunto de contêineres, sendo somente um deles o contêiner principal *(Main-Container)*, dotado de dois agentes particulares: o Sistema Gerenciador de Agentes *(Agent Management System - AMS)* e o Diretório Facilitador *(Directory Facilitator - DF)* [20].

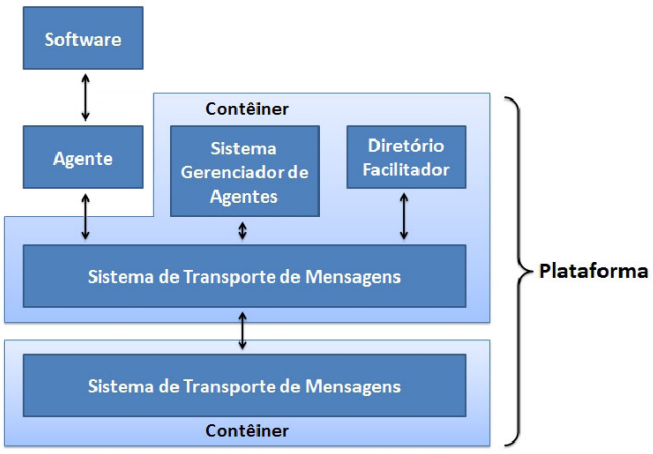


Figura 6.1: *Arquitetura do JADE.*

O *AMS* supervisiona a plataforma por completo, controlando o ciclo de vida de todos os agentes. Além disso, através de um serviço particular denominado serviço de páginas brancas *(white-pages)*, o *AMS* mantém os estados e identificadores dos agentes, que devem então se registrar nele para obter um *AID (Agent Identifier)* válido.

O *DF* é responsável pelo serviço de páginas amarelas *(yellow-pages)*, usado por qualquer agente que queira registrar ou procurar serviços disponíveis. A notificação dos agentes sobre um novo serviço registrado ou modificado também é realizada pelo *DF*.

Os agentes se comunicam através do Sistema de Transporte de Mensagens *(Messaging Management System - MMS)*, também chamado de Canal de Comunicação de Agentes *(Agent Communication Channel - ACC)*. O *MMS* é responsável por toda comunicação entre agentes dentro e fora da plataforma, já que o agente pode se relacionar tanto com a aplicação externa quanto com o JADE [7].

O transporte de mensagens em um mesmo contêiner é realizado pelos métodos Java. Já o transporte de mensagens entre contêineres é viabilizado pela *JRM (Java Remote Message)*. Fechando o ciclo, o transporte entre plataformas pode ser feito pelo protocolo *IIOP (Internet Inter-Orb Protocol)* [7].

6.1.2 Ciclo de Vida de Agentes

Os estados possíveis de um agente no JADE, que definem seu ciclo de vida, é a implementação do modelo especificado pela FIPA, conforme ilustrado na Figura 6.2 [19].

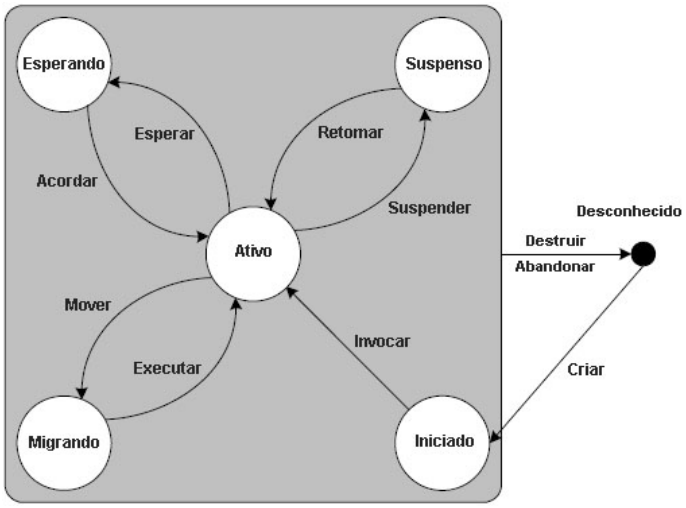


Figura 6.2: *Ciclo de Vida dos Agentes.*

Estes estados são representados como constantes estáticas da classe *Agent*, podendo ser modificados através de métodos específicos. Os principais estados são [19]:

- **INITIATED:** o agente foi instanciado mas ainda não se registrou no *AMS*. Desta forma ele ainda não possui nome e endereço, o que inviabiliza sua comunicação com outros agentes;

- **ACTIVE:** o agente está registrado no *AMS*, portanto possui nome e endereço, além de possuir todas as funcionalidades de um agente;

- **SUSPENDED:** o agente está interrompido, assim sua *thread* permanece suspensa e nenhum comportamento é executado;

- **WAITING:** o agente está bloqueado, esperando por determinada ocorrência. Normalmente um agente é acordado através de uma mensagem;

- **DELETED:** o agente está definitivamente excluído, já que sua *thread* encerrou a execução, desvinculando-o automaticamente do *AMS*;

- **TRANSIT:** o agente está migrando para uma nova localização.

6.1.3 Tarefas de Agentes

As tarefas no JADE representam as ações a serem realizadas por um agente, obtendo a denominação de comportamentos *(behaviours)*. Cada agente JADE ocupa uma *thread* separada, que por sua vez, também é compartilhada por todas as suas tarefas.

Embora possam ser criadas tarefas que utilizem *threads* separadas, este recurso não é visto como uma boa prática, pois a mudança de *threads* é 100 vezes mais lenta se comparada a uma chamada de método, além de prover problemas de sincronização [102].

O escalonador de tarefas fica escondido do programador, realizando um agendamento não preemptivo colaborativo *round-robin* [20]. Isto significa que cada execução de tarefa corresponde a uma única fase ativa, tornando-a ininterrupta enquanto recebe os recursos de processamento.

A seguir é feita uma descrição das classes que implementam *behaviours* [7]:

- **Behaviour:** superclasse de todas as classes que implementam tarefas, possuindo a estrutura dos comportamentos dos agentes JADE;

- **SimpleBehaviour:** modela tarefas atômicas, ou seja, aquelas que são realizadas inteiramente sem nenhuma interrupção;

- **OneShotBehaviour:** modela tarefas atômicas que não podem ser bloqueadas e só podem ser executadas uma única vez;

- **SenderBehaviour:** implementa a classe *OneShotBehaviour* para envio de mensagens *ACL (Agent Communication Language)*;

- **ReceiverBehaviour:** implementa uma tarefa para recebimento de mensagens *ACL*;

- **CyclicBehaviour:** modela tarefas atômicas que devem ser executadas ininterruptamente;

- **WakerBehaviour:** implementa um *OneShotBehaviour* que é executado somente após um tempo em milissegundos transcorrido;

- **TickerBehaviour:** implementa um *CyclicBehaviour* que deve ser executado periodicamente;
- **CompositeBehaviour:** constitui-se por outros *behaviours* (filhos), que controlam sua execução;
- **SequentialBehaviour:** executa suas subtarefas sequencialmente, enquanto alguma delas ainda não foi finalizada;
- **ParallelBehaviour:** executa suas subtarefas concorrentemente e termina quando uma condição particular nas mesmas ocorre;
- **FSMBehaviour:** executa suas subtarefas de acordo com uma máquina de estados finitos.

6.1.4 Comunicação entre Agentes

A comunicação entre os agentes no JADE é viabilizada através do transporte de mensagens *ACL (Agent Communication Language)*. Para que seja possível o envio destas mensagens, inicialmente é necessário instanciar um objeto da classe *ACLMessage* que as representem. Em seguida, campos deste objeto como o destinatário, o conteúdo da mensagem, além de opções como ontologias, devem ser corretamente preenchidos, para que finalmente seu envio possa ser realizado por um método específico [7].

Antes da criação do objeto da classe *ACLMessage* é necessário criar um objeto da classe *AID* que representará o identificador do agente destinatário, que ao ser instanciado, recebe como parâmetro o endereço de destino da mensagem a ser enviada [7].

Após esse processo, é possível então criar o objeto da classe *ACLMessage*, passando como parâmetro ao construtor a performativa da mensagem. Esta performativa, também chamada de ato de comunicação, indica ao receptor a intenção da mensagem, podendo ser identificadas como: *Agree*, *Cancel*, *Confirm*, *Failure*, *Inform*, *Not Understood*, *Propose*, *Refuse*, *Request*, dentre outras [7].

Todas essas performativas são baseadas na teoria do ato da fala que define funções da comunicação, e que objetivam classificar as mensagens segundo a função que exercem, como *Interrogativas*, *Exercitivas*, *Referenciais*, *Fáticas*, etc [96].

Para que um agente possa receber mensagens, o passo inicial é a criação de uma tarefa capaz de perceber as mesmas. Em seguida, torna-se necessário a construção de uma instância do conteúdo da mensagem através de um método específico da plataforma. Finalmente, basta processá-la e, se necessário, retornar alguma resposta [7].

O JADE também disponibiliza o serviço de páginas amarelas, pelo qual os agentes podem fornecer e procurar por serviços. Em um ambiente heterogêneo, com agentes especializados em tarefas específicas, é fundamental este tipo de recurso cooperativo.

CAPÍTULO 7

Frameworks Gráficos Java

O modelo de referência para o processo de visualização de informações apresentado por Card, ilustrado na Figura 5.4, compõe-se de três etapas: as Transformações de Dados, o Mapeamento Visual e as Transformações Visuais. As duas últimas etapas deste modelo são fatores determinantes para o propósito da elaboração visual dos dados, uma vez que a imagem final pode simplificar informações ou revelar padrões ocultos.

Para Lima, o Mapeamento Visual possibilita que os dados finalmente ganhem vida através de uma forma gráfica deliberada [68]. Ele leva em consideração fatores importantes como cor, contraste, espaçamento, posição, tamanho e forma, contemplando não apenas módulos individuais, mas também a composição de todo o ambiente contíguo.

Já as Transformações Visuais representam uma ferramenta de descoberta e interatividade, fornecendo a camada final de coalescência para exploração. Esta camada de unificação do modelo é fundamental para a análise exploratória, permitindo aos usuários investigar, filtrar, manipular, remodelar e examinar o resultado visual, a fim de identificar propriedades, relações, regularidades, ou padrões nos dados referentes.

Em uma definição ampla de visualização, é consensual que a informação pode ser transmitida com sucesso através de execuções estáticas ou interativas. No entanto, é relevante o questionamento do que realmente define Visualização da Informação para além de outros campos paralelos, como Design de Informação ou Gráficos de Informação. De fato, a sua natureza suportada por computador interativo é o que realmente a torna diferente, perante o alto grau de complexidade dos sistemas analisados [68].

Por essa razão, a escolha de determinado *framework* para a criação do Mapeamento Visual, bem como para a viabilização das Transformações Visuais, torna-se preponderante para que análises mais profundas venham a ser realizadas. Tais *frameworks* irão fornecer os meios para a criação de um sistema de visualização de informação, e como tal, devem proporcionar alto nível de personalização e uma maneira fácil de usar os recursos gráficos do *hardware*.

Pode-se distinguir dois tipos principais de *frameworks* gráficos: de alto e baixo nível. Cada um fornece diferentes níveis de controle sobre o *hardware* gráfico de uma máquina. Enquanto os primeiros podem proporcionar uma maior gama de recursos

visuais, os segundos podem produzir uma renderização mais rápida ou um controle mais efetivo sobre o que está sendo desenhado na tela [34].

Não obstante, em se tratando de redes complexas, a utilização de *frameworks* de alto nível é fundamental, pois além de usufruírem de um forte poder de abstração, disponibilizam vastos recursos interacionais, geralmente ausentes em seus contrapartes.

Diversos *frameworks* pertencentes a esta classe foram analisados durante a construção desta dissertação como: *JUNG* [78], *Prefuse* [58], *JavaView* [85], *Piccolo* [18], *yFiles* [110], e *Gephi* [47]. De forma geral, os *frameworks* JUNG e Prefuse destacam-se na academia como os mais utilizados em projetos com grafos que representam redes complexas, como pode ser observado nestes trabalhos [34] [5] [21] [54] [93].

Escrito em Java, com versão e documentação atualizados, e possuidor de diversos recursos para mapeamento e transformações visuais, além de disponibilizar algoritmos para criação e análise de grafos, o JUNG foi a ferramenta mais completa para os propósitos desta obra, sendo consequentemente a escolha mais prudente.

7.1 JUNG - *Java Universal Network/Graph Framework*

JUNG é um *framework* de código aberto extensível, desenvolvido e mantido desde 2003 para permitir a modelagem, análise e visualização de dados passíveis de serem representados na forma de grafos [79], podendo ser obtido atualmente em [78].

Fornecido livremente sob a licença de código aberto *BSD (Berkeley Software Distribution)* [100], o JUNG permite que qualquer pessoa crie obras derivadas de suas bibliotecas, desde que elas reconheçam a contribuição do *framework* para o seu trabalho. Como uma biblioteca, o JUNG pode ser utilizado tanto para a construção de ferramentas orientadas por redes, como para fornecer tal orientação para sistemas existentes.

A natureza de código aberto do projeto facilita os processos de criação de códigos, e de compreensão do funcionamento da ferramenta. Como resultado, os membros da comunidade do JUNG têm sido capazes de contribuir com um grande número de extensões, algoritmos e correções relevantes.

As principais características do JUNG incluem [79]:

- Suporte para diversas representações de entidades e suas relações, incluindo grafos orientados e não orientados, grafos multimodais, multigrafos e hipergrafos;

- Mecanismos para anotar grafos, entidades e relações com metadados. Estas capacidades facilitam a criação de ferramentas de análise de conjuntos de dados complexos que podem examinar as relações entre as entidades, bem como os metadados anexados a cada entidade e relação;

- Implementação de uma série de algoritmos da Teoria dos Grafos, análise exploratória de dados, análise de redes sociais, e aprendizagem de máquina. Estes incluem rotinas para *clustering*, otimização de decomposição, geração de grafos aleatórios, análise estatística e cálculo de distâncias da rede, fluxos, e as medidas de classificação (centralidade, *PageRank*, *HITS*, etc);
- Recursos de visualização que facilitam a construção de ferramentas para a exploração interativa dos dados da rede. Os usuários podem escolher entre os *layouts* fornecidos e os algoritmos de visualização, ou usar a estrutura para criar seus próprios algoritmos personalizados;
- Mecanismos de filtragem que extraem subconjuntos de uma rede, permitindo aos usuários centralizarem sua atenção em porções específicas da mesma.

Esses recursos fazem do JUNG uma excelente plataforma para análise exploratória de dados relacionais. Ele pode ser utilizado em fragmentos simples de código em teste, ou para auxiliar no desenvolvimento de uma ferramenta sofisticada, com uma interface gráfica de usuário. O JUNG não é por si só uma ferramenta autônoma, mas sim uma biblioteca que pode ser utilizada para apoiar a construção de ferramentas específicas.

7.1.1 Arquitetura do JUNG

Uma importante característica do JUNG é o seu modelo de arquitetura, apresentado na Figura 7.1 [79].

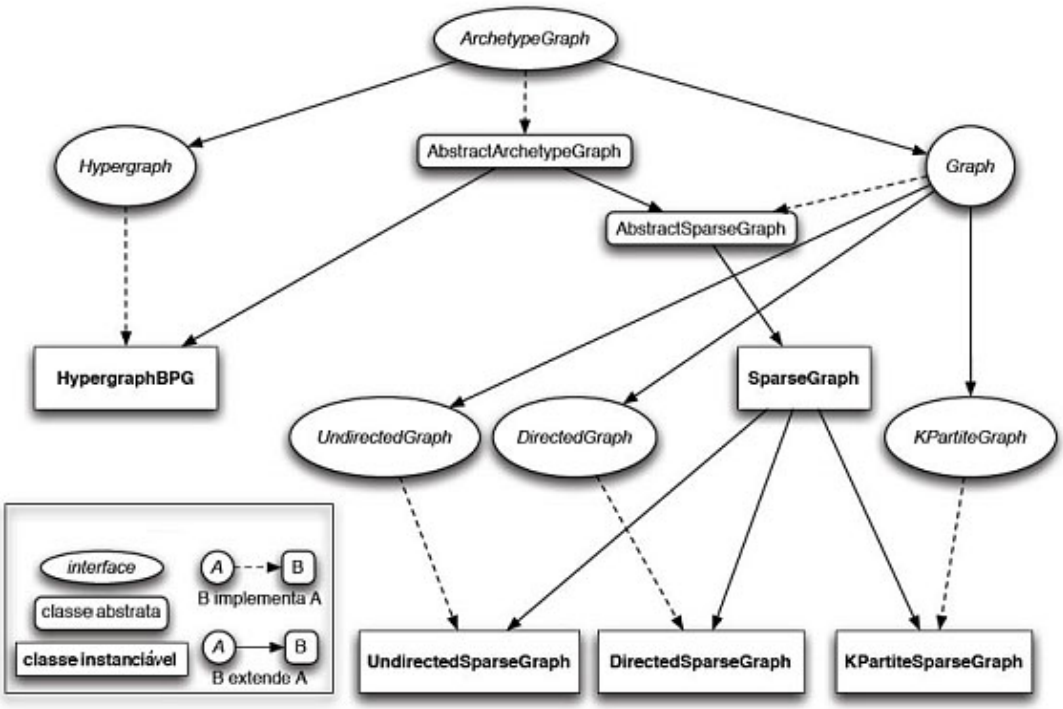


Figura 7.1: *Arquitetura do JUNG.*

A biblioteca foi criada de modo a abranger as mais diversas necessidades, sendo assim bastante genérica, capaz de representar grafos de diversas formas [84].

No nível mais elevado, existe a interface denominada *ArchetypeGraph* que define as diretrizes dos grafos. No nível imediatamente inferior, especializando a interface *ArchetypeGraph*, encontra-se a interface *Graph*. Esta objetiva representar elementos de grafos sem arestas paralelas.

Para se trabalhar com grafos direcionados e não direcionados, distinguindo-os dos demais, foram criadas duas interfaces: *DirectedGraph* e *UndirectedGraph*. Estas interfaces permitem validações em tempo de compilação, não permitindo referências a elementos fora de seus domínios [84]. A seguir são disponibilizadas as implementações destas interfaces, através das classes *UndirectedSparseGraph* para grafos não direcionados, e *DirectedSparseGraph* para grafos direcionados.

Em uma camada intermediária, existe uma classe implementando funcionalidades comuns a grafos direcionados e não direcionados denominada *AbstractSparseGraph*.

Todas essas implementações utilizam a representação de grafos em forma de listas de adjacência, o que pode ser observado através dos próprios nomes das classes que contêm a palavra *Sparse* [84]. A justificativa é que grafos esparsos, cujo número de arestas é da ordem de $O(n)$, são em geral representados de maneira mais eficiente e econômica usando este tipo de estrutura de dados. Já os grafos densos, que possuem a quantidade de arestas da ordem de $O(n^2)$, são melhor projetados com matrizes de adjacência.

7.1.2 Criação de Grafos

Existem três formas para a criação de grafos no JUNG. A primeira pode ser feita a partir da leitura de dados em um arquivo. O *framework* suporta o formato *Pajek*, baseado em uma estrutura de arquivo texto, e o formato *GraphML*, baseado em arquivos *XML* [79].

A segunda maneira de gerar um grafo no *framework* é através da criação individual dos vértices e das arestas [79]. Para isto, anteriormente deve ser criada uma instância das classes *UndirectedSpareGraph* ou *DirectedSparseGraph* que determinará o objeto referente ao grafo. Em seguida, vértices e arestas devem ser adicionados sequencialmente, com a restrição de arestas só podendo ser criadas entre dois vértices existentes.

Um grande diferencial da ferramenta refere-se ao tipo de dado que pode representar um vértice ou uma aresta. No JUNG, qualquer objeto válido pode fazer esta representação, o que viabiliza a utilização de classes específicas como estrutura de cada um destes elementos do grafo. Com este recurso, aumenta-se enormemente a flexibilidade, principalmente quando se trata de armazenar e recuperar os atributos desses elementos.

Uma última forma de produzir um grafo no *framework* é através de algoritmos preexistentes. *Small-World Graphs*, *Power-Law Graphs*, *Random Graphs* [78], são alguns

dos exemplos de algoritmos implementados na plataforma. Consequentemente, o usuário precisa apenas de informar os respetivos parâmetros de cada algoritmo e processá-lo.

7.1.3 Visualização de Grafos

Para visualizar um grafo no JUNG são necessárias três implementações consecutivas. Primeiro, cria-se a estrutura do grafo no código. Em seguida, um *layout* para a organização dos vértices e arestas deve ser vinculado ao grafo. Finalmente, um componente para exibir o grafo com o *layout* anteriormente definido deve ser produzido [88].

Na visualização e renderização do grafo são utilizadas as bibliotecas *Swing* e *Awt* do Java. A *Swing* é responsável pela criação dos componentes gráficos, e a *Awt* é utilizada para criação de interfaces com o usuário e para a pintura de gráficos e imagens [79].

Na Figura 7.2 é apresentado um grafo não direcionado criado pelo *framework* JUNG, com sua configuração padrão de visualização.

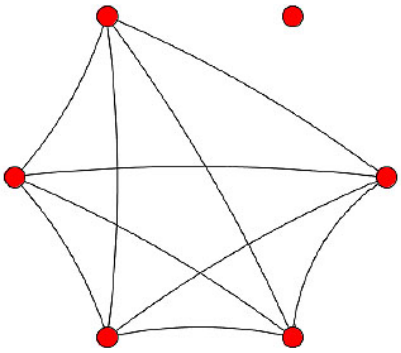


Figura 7.2: *Grafo Gerado pelo JUNG.*

A disposição dos vértices e arestas no JUNG é gerada automaticamente através do uso de *layouts* pré-definidos. A Figura 7.2 demonstrou o uso do *layout KKLayout*. Além deste, existem outros *layouts* fornecidos pelo *framework* como: *ISOMLayout*, *CircleLayout*, *FRLayout* e *SpringLayout* [79], detalhados a seguir:

- **KKLayout:** (Algoritmo: Kamada-Kawai) - Usa um *layout* de nó, não respeitando as chamadas de filtro;
- **ISOMLayout:** (Mapa de Auto-Organização Meyer) - Implementa um algoritmo que realiza uma auto-organização de *layout*;
- **CircleLayout:** (Layout simples de posições randômicas de vértices em um círculo) - Posiciona os vértices igualmente espaçados em um círculo regular;
- **FRLayout:** (Algoritmo: Fruchterman-Reingold) - A disposição dos nós segue um algoritmo baseado na concepção de que nós próximos acabam por repelir-se;
- **SpringLayout:** Possui nós filhos associados com determinados tipo de regras.

7.1.4 Filtros

O JUNG pode suportar a utilização de filtros, através dos quais é possível converter um grafo em algum de seus subgrafos [79]. Os vértices e as arestas podem ser filtrados de acordo com um predicado especificado, valendo-se de algum atributo referente a estes elementos. Isso permite aos usuários filtrar sem a necessidade de criar novos grafos menores, poupando logicamente, tempo e recursos computacionais.

Embora o usuário possa escrever um filtro personalizado apenas pela implementação da interface *Filter*, geralmente torna-se mais fácil estender uma das classes base de filtro fornecidas pela plataforma. Para o primeiro caso é necessário a reescrita do método booleano que define o *status* de visualização de cada vértice ou aresta do grafo.

7.1.5 Algoritmos

Diversos algoritmos de rede são disponibilizados pela plataforma, sendo suas aplicações específicas para cada categoria.

Os algoritmos de aglomeração ou *clustering* se baseiam nos conjuntos de objetos semelhantes de alguma forma. Em uma rede, a semelhança é muitas vezes verificada por propriedades topológicas como a conectividade, mas também pode ser identificada pelas propriedades dos vértices ou arestas. Um destes algoritmos de agrupamento é o *EdgeBetweennessClusterer*, que calcula aglomerados para um grafo com base na propriedade de intermediação das arestas [95], como visualizado na Figura 7.3 [80].

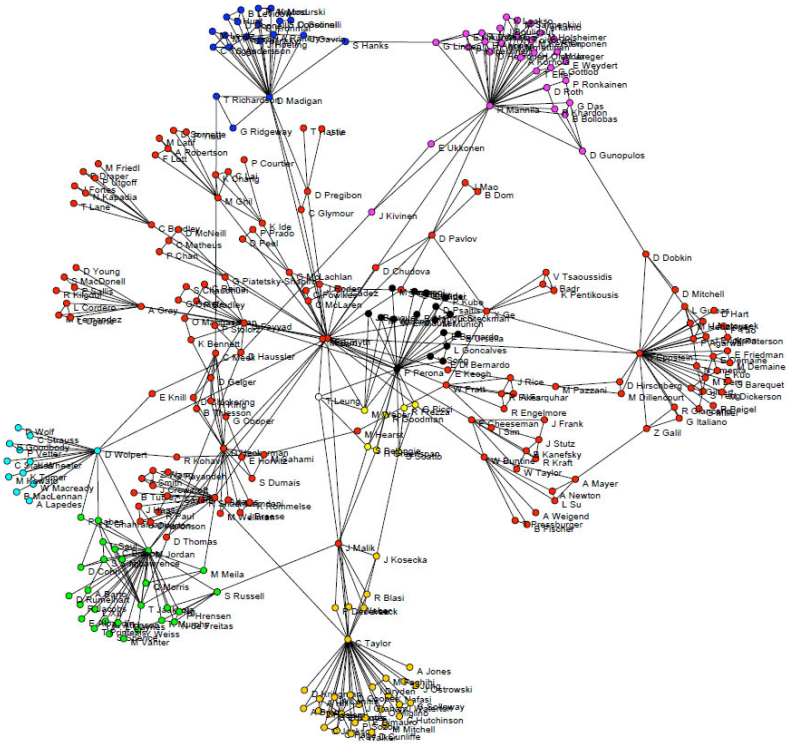


Figura 7.3: *Algoritmo EdgeBetweennessClusterer.*

Os algoritmos topológicos, de caminhos e fluxos, executam operações em grafos relacionadas com sua topologia. Um exemplo importante desta categoria é o *DijkstraShortestPath*, que calcula o comprimento do caminho mínimo ponderado a partir dos vértices origem e destino especificados [95]. A Figura 7.4 [80] exemplifica esta técnica.

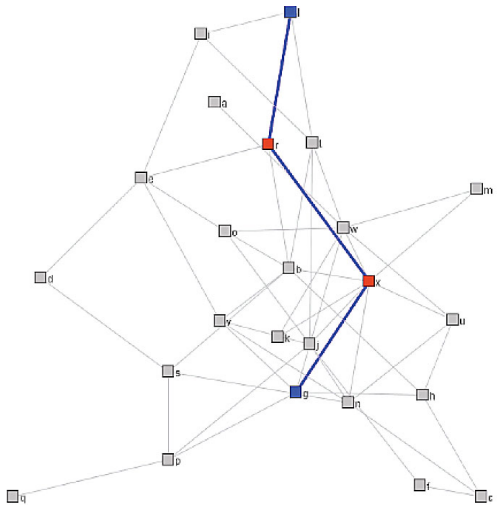


Figura 7.4: *Algoritmo DijkstraShortestPath.*

Outra categoria contemplada pelo *framework* são os algoritmos de importância, que avaliam quão relevante é cada vértice ou aresta para a rede, de acordo com critérios geralmente baseados no posicionamento deles perante o resto do grafo. Dentre estes algoritmos inclui-se o *BetweennessCentrality*, que rotula cada vértice e aresta de um grafo com um valor obtido a partir do número de caminhos mínimos que passam por eles [95].

Algoritmos de medidas estatísticas referentes aos grafos também são fornecidos pelo JUNG. Alguns contemplados são: coeficiente de aglomeração, diâmetro da rede, distância geodésica média e grau de distribuição [95].

7.1.6 Recursos de Interação

Os recursos de interação providos pelo JUNG destacam-se por sua grande contribuição ao processo de Transformações Visuais. Adição, remoção, seleção e movimentação de vértices e arestas, coloração, mudança de forma, rotação do eixo da imagem, deformação, aumento e diminuição da escala *(zoom)*, são alguns dos recursos interacionais disponibilizados pelo *framework*.

Combinados com os *layouts* de visualização, tais recursos tornam-se imprescindíveis para a análise dos grafos gerados. A possibilidade de interagir com o mesmo grafo em diferentes visualizações, sendo estas ainda passíveis de deformações deliberadas, aumenta a chance de descoberta de padrões despercebidos.

A Figura 7.5 mostra em seu lado esquerdo a configuração inicial de um grafo gerado por Santos no JUNG com o *layout SpringLayout2*, utilizando-se de determinada

base de dados [88]. O lado direito corresponde à estabilização do *layout* promovido automaticamente pela plataforma.

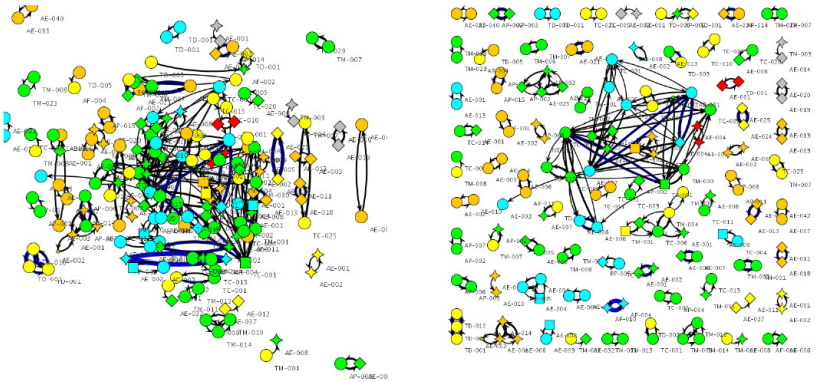


Figura 7.5: *Grafo Inicial e Estabilizado.*

Ambos os lados da Figura 7.6 mostram diferentes ampliações do grafo da figura anterior *(zoom)*.

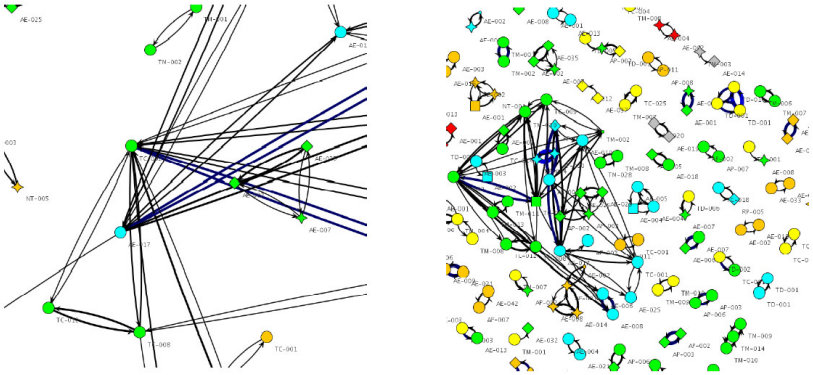


Figura 7.6: *Grafo Ampliado.*

O lado esquerdo da Figura 7.7 refere-se à rotação do eixo da imagem preservando as orientações de seus rótulos. O lado direito corresponde à inclinação da imagem para se obter uma nova perspectiva.

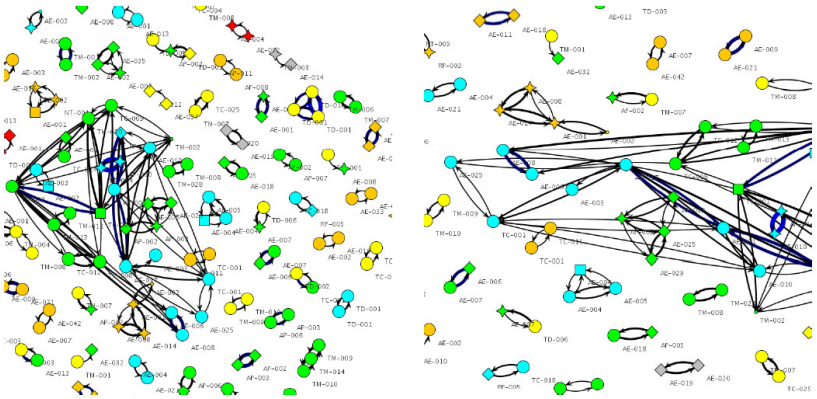


Figura 7.7: *Grafo Rotacionado e Distorcido.*

CAPÍTULO 8

Análise de Trabalhos Correlatos

Neste capítulo, uma breve descrição de alguns dos trabalhos relacionados com o propósito desta obra é apresentada. *The Oracle of Kevin Bacon*, *Vizster*, *SocNetV*, *Facebook Visualiser*, *TagGraph* e *MentionMapp* são as aplicações analisadas, que possuem características semelhantes às da aplicação Fluzz. Algumas delas trabalham a questão visual, dispondo de recursos gráficos para a representação de redes. Outras, além deste último recurso, se propõem a disponibilizar algoritmos de busca, e algoritmos para a criação e análise de redes sociais, limitando-se, porém, aos modelos já existentes na literatura.

Uma excelente fonte de pesquisa sobre outros projetos desta natureza pode ser encontrada no *site VisualComplexity.com* [67]. Idealizado pelo pesquisador da *Microsoft*, Manoel Lima, este *site* foi criado com o intuito de ser um espaço unificado para todos os interessados na questão de visualização de redes complexas.

Nenhum dos projetos analisados pertence à iniciativa brasileira, demonstrando que a pesquisa nacional nesta área recente, ainda carece de uma atenção significativa.

8.1 The Oracle of Kevin Bacon

Não é por acaso que este foi o primeiro trabalho correlato à pesquisa a ser analisado. Pouco tempo depois que os estudantes de computação Glen Wasson e Brett Tjaden disponibilizaram o *site The Oracle of Kevin Bacon* [105], Duncan Watts e Steven Strogatz obtiveram sua primeira confirmação real da existência de redes de mundo pequeno. O mesmo *site* também serviu para Albert Barabási verificar a existência dos *hubs*. Ironicamente, os possíveis padrões estruturais ocultos da natureza foram descobertos pelos cientistas, através de uma aplicação originada por uma simples brincadeira.

No início de 1994, um jogo conhecido por *Jogo do Kevin Bacon* nasceu na cultura popular americana. Alunos do Albright College, na Pensilvânia, imaginaram que como o ator Kevin Bacon tinha atuado em filmes de naturezas tão diversas, isso tornaria possível conectá-lo a quase qualquer ator em *Hollywood* [14].

O objetivo do jogo é determinar o número Bacon de um determinado ator descobrindo o caminho mais curto até o astro do cinema. Alguém que já atuou em um

filme com Bacon possui um número Bacon 1. Caso uma pessoa não tenha atuado com ele, mas atuou com alguém que já o fez, esta pessoa possui um número Bacon 2.

Graças a este jogo, em 1997, Wasson e Tjaden perceberam que a determinação da distância entre dois atores quaisquer era um projeto viável da Ciência da Computação, caso fosse possível obter acesso a um banco de dados dos atores e filmes já lançados [14].

Após obterem acesso ao *Internet Movie Database* [9], maior fonte cinéfila mundial, transcorreu pouco tempo para que Wasson e Tjaden instalassem o *site The Oracle of Kevin Bacon*. Ao informar os nomes de dois atores quaisquer, em milissegundos o *site* apresenta o caminho mais curto entre eles, listando a cadeia de filmes e atores pelos quais são conectados. Um exemplo é ilustrado na Figura 8.1, onde foi calculado o caminho mínimo entre o ator brasileiro Antônio Fagundes, e o ator americano Tom Cruise.

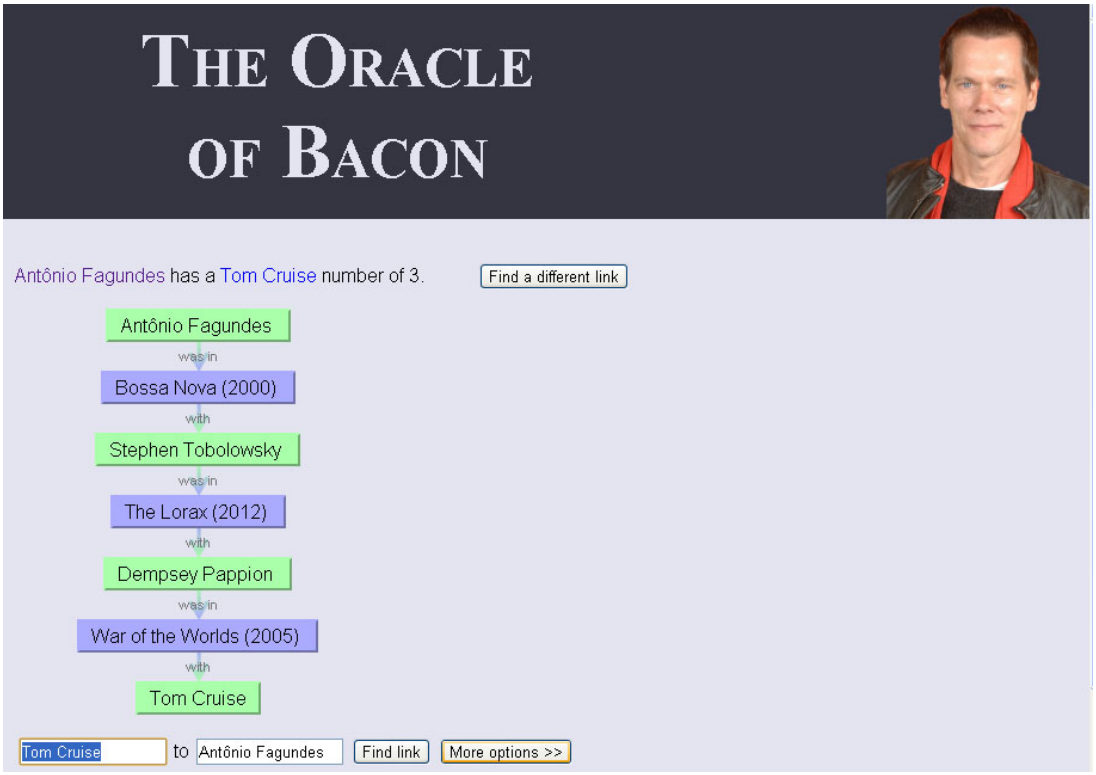


Figura 8.1: *The Oracle of Kevin Bacon.*

A cada duas semanas o *site* realiza atualizações de vários arquivos de banco de dados do *Internet Movie Database*, e atualmente contempla cerca de 1,5 milhões de atores e atrizes, e 1,2 milhões de filmes e programas de TV [105].

A aplicação usa uma busca em largura para encontrar o caminho mínimo entre pares de atores, uma vez que o grafo representativo da rede não é ponderado, o que a diferencia drasticamente da aplicação Fluzz. Além disso, diferentemente do Fluzz, o *site The Oracle of Kevin Bacon* não possui recursos para visualização gráfica de parte ou de toda a rede de atores, concentrando-se somente na representação em formato

linear dos caminhos encontrados pelo mecanismo de busca. Esta limitação tornou inútil a disponibilização de recursos interativos pela ferramenta, que obviamente não os possui.

Sempre que o *site* responde a uma consulta, os resultados são armazenados em *cache* para que futuras solicitações referentes ao mesmo ator possam ocorrer mais rapidamente. Cerca de 80% das consultas são atendidas por este recurso [105].

O *The Oracle of Kevin Bacon*, inconscientemente ao seu propósito original, se tornou um marco na pesquisa sobre redes. Importantes características identificadas neste meio, como aglomeração, distâncias pequenas, componentes gigantes e *hubs* revelaram que o mundo não é formado por eventos aleatórios, como acreditavam Erdös e Rényi.

8.2 Vizster

O Vizster constitui um ambiente visual para a exploração e análise de redes sociais, incluindo dados de perfil e topológicos. A escala de exibição de informação e o *layout* foram escolhidos para apoiar o comportamento e a capacidade observadora dos usuários, permitindo-lhes expandir as redes visualizadas mantendo marcações [56].

Realce interativo é usado para explorar as relações de amizade na plataforma. Panorâmica, *zoom* e técnicas de distorção são fornecidos para ajudar os usuários a navegar pela visualização da rede. Buscas textuais e um atributo de visualização interativa conhecido por *X-ray mode* permitem exploração visual de dados de perfil dos membros. Finalmente, uma análise de comunidades é fornecida visualmente para ajudar os usuários a construir e explorar estruturas de alto nível destas organizações.

A ideia para a construção do Vizster se apoiou fortemente nos aspectos exploratórios e divertidos de uma importante rede social *online*, o Friendster. Não obstante, enquanto os usuários regularmente exploravam a rede no Friendster, os formatos lineares limitavam tais explorações [56]. Isto levou Jeffrey Heer e Danah boyd a desenvolver ferramentas mais ricas de exploração de redes.

Para os autores do projeto, o uso de imagens foi indispensável para a identificação de pessoas. Além disso, eles perceberam a importância de disponibilizar recursos de pesquisa para os dados de perfil, trabalhando os resultados visualmente na interface [56].

O Vizster apresenta redes sociais usando uma representação familiar *nó-link*, onde os nós representam as pessoas, identificadas por seus respectivos nomes e fotografias, e as ligações representam os laços de amizade, como mostrado na Figura 8.2 [57].

As redes são apresentadas como redes egocêntricas: redes que consistem de um indivíduo e o seus amigos mais próximos. Usuários podem expandir a exibição selecionando nós, para fazer os contatos imediatos dos amigos visíveis também. À direita do visor da aplicação existe um painel de apresentação do perfil das pessoas, onde se situa uma ferramenta de manipulação de pesquisas sobre o texto do mesmo.

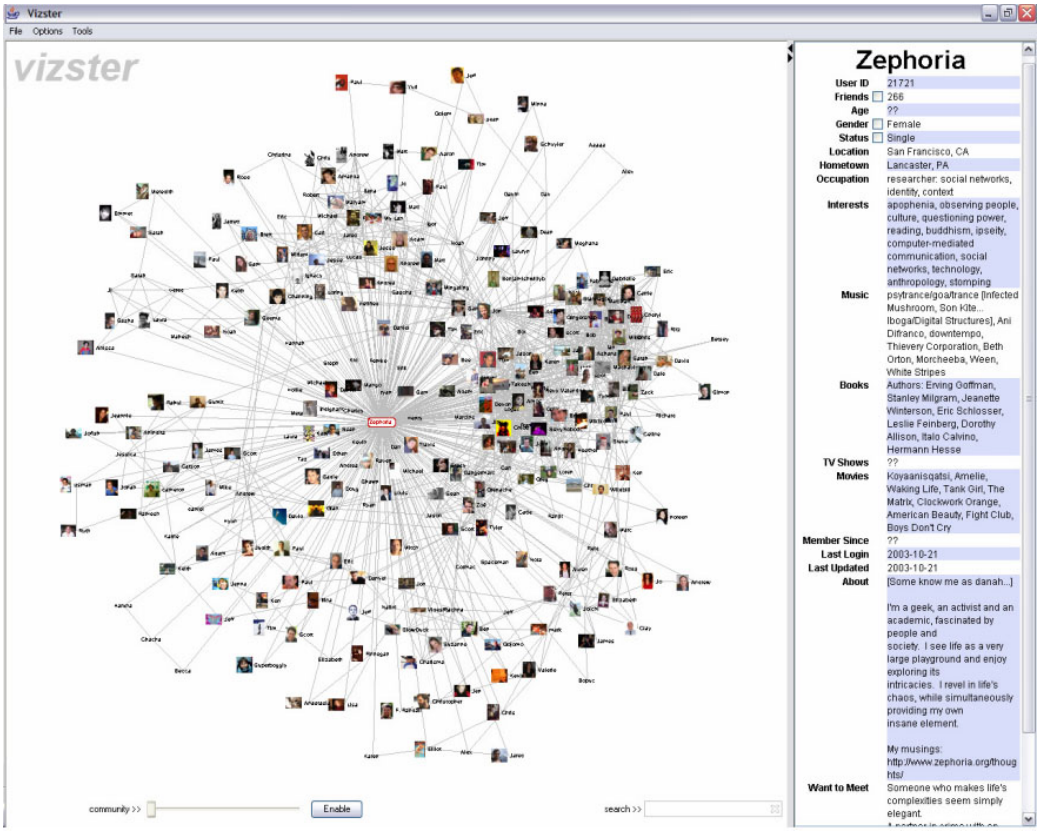


Figura 8.2: *Vizster.*

Dada a falta de familiaridade a priori do usuário com sua rede estendida, uma perspectiva egocêntrica não só carrega um fardo menos perceptivo e computacional, mas garante a presença de pontos de referência facilmente identificáveis para orientar o usuário, que se optar, pode expandir gradativamente a rede [56].

Diferentemente do Fluzz, o Vizster não disponibiliza diferentes *layouts* de visualização para as redes. Também não possibilita a localização de caminhos entre pares de nós, já que o foco da aplicação é simplesmente a representação gráfica de redes sociais.

O Vizster foi escrito em Java usando o *framework* gráfico Prefuse [58]. Para apoiar a procura de palavras-chave, o motor de busca Lucene [97] foi integrado à aplicação, que persiste seus dados no banco de dados MySQL [82] ou em arquivos XML. A aplicação pode ser obtida livremente no endereço [57].

8.3 SocNetV

O SocNetV, acrônimo de *Social Network Visualizer*, é uma ferramenta flexível e de fácil utilização para a análise e visualização de redes sociais. Ele permite a construção e exploração de redes carregando arquivos em vários formatos (GraphViz, GraphML, adjacência, Pajek, UCINET, etc.), bem como suas modificações [13]. A aplicação permite ao

usuário calcular propriedades básicas de rede, tais como densidade, diâmetro, distâncias, comprimentos de caminho mínimo, centralidades e coeficientes de aglomeração.

Além disso, o SocNetV oferece um rastreador Web embutido *(crawler)*, permitindo que o usuário crie automaticamente a rede de todos os *links* encontrados em um determinada *URL* inicial. Dispõe ainda de vários algoritmos de *layout* para visualizações de redes, e também para geração das mesmas como Erdös-Renyi, Watts-Strogatz, etc [62].

Desenvolvido por Dimitris Kalamaras em C ++ e Qt [76], um *framework* gráfico de código aberto criado pela *Nokia*, o *software* está licenciado sob a *GNU General Public License 3 (GPL3)*, e pode ser obtido no endereço [62]. Uma ilustração da interface gráfica do *software* é apresentada na Figura 8.3 [62].

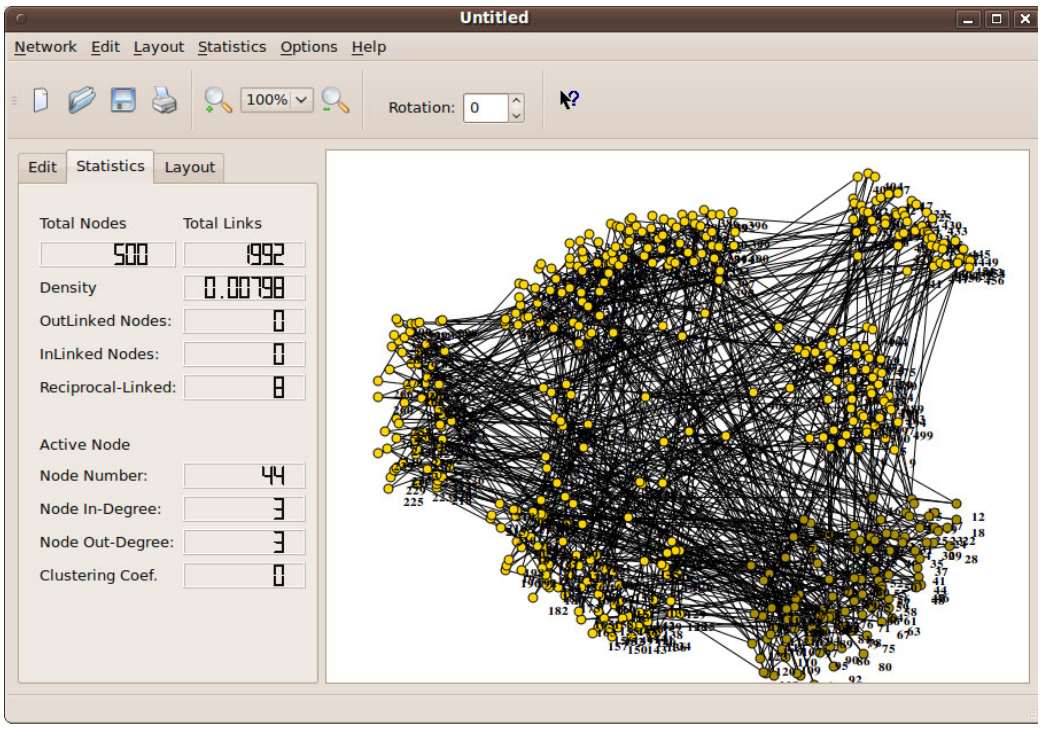


Figura 8.3: *SocNetV.*

Basicamente, o SocNetV é a aplicação mais próxima do Fluzz, diferenciando-se por algumas particularidades. Primeiro, não existem modelos avaliativos dos laços de amizade, o que impossibilita a construção de um grafo ponderado representativo das redes sociais. A utilização de agentes colaborativos heterogêneos para a localização de caminhos entre pares de nós, é outro diferencial do Fluzz perante o SocNetV, que também não possui um modelo próprio para a evolução deste tipo de rede.

8.4 Facebook Visualiser, TagGraph, Mentionmapp

Esta subseção foi destinada à apresentação de aplicações desenvolvidas para *sites* específicos, e que basicamente realizam o mesmo trabalho visual e operacional,

distinguindo-se somente por suas fontes informacionais. Diferentemente do Fluzz, nenhum deles oferece mecanismos para a localização de caminhos nas redes, nem possibilita a manipulação de redes fora de seus domínios.

O Facebook Visualiser [101] é uma ferramenta criada por Sebastian Van Sande para explorar graficamente o Facebook. O usuário pode utilizá-lo para ver seus relacionamentos, e para verificar como seus contatos estão conectados entre si. A ferramenta, apresentada à esquerda da Figura 8.4 [67], provê pequenos recursos, como a visualização da fotografia importada do Facebook, e filtros por sexo e situação de relacionamento.

Criado por Forrest Oliphant, o TagGraph [77], assim como o Facebook Visualiser, pode mapear graficamente a rede egocêntrica do usuário no Facebook. Porém, sua concepção inicial foi devido ao relacionamento entre *tags* do *site* de imagens Flickr [109].

O aplicativo, apresentado no centro da Figura 8.4 [67], cria um grafo que viabiliza a navegação entre as *tags* criadas pelos usuários no Flickr, mapeando a rede de imagens relativas à determinada *tag*. Caso o usuário clique em uma das imagens visualizadas, as mesmas são carregadas diretamente no Flickr. Alguns recursos são providos pela aplicação como *zoom* e movimentação dos objetos.

Lançado pela Asterisq, o Mentionmap [72] é uma aplicação Web para explorar a rede social *online* Twitter [44]. Ele permite que o usuário descubra as pessoas com que ele mais interage, carregando as atualizações de cada usuário do Twitter *(tweets)* e encontrando as pessoas e *hashtags* mais utilizadas pelos indivíduos a eles conectados.

Clicando em um usuário, o Mentionmap, apresentado à direita da Figura 8.4 [67], mostra a sua rede de menções, bem como detalhes de seu perfil. As linhas traçadas entre os nós tornam-se mais espessas à medida que mais menções são realizadas, e pairando sobre uma aresta é possível verificar o número de menções. O Mentionmap também provê um recurso básico de pesquisa de usuários, através da digitação de seus nomes.

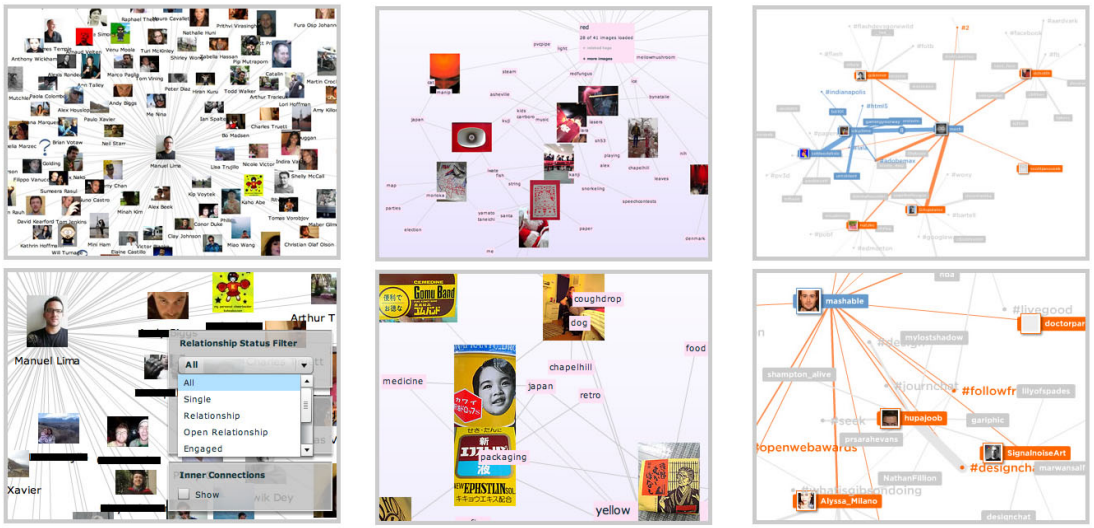


Figura 8.4: *Facebook Visualiser, TagGraph, Mentionmapp.*

CAPÍTULO 9

Projeto da Aplicação

Ao término dos estudos descritos anteriormente, uma nova e importante fase foi iniciada: a concepção do projeto da aplicação Fluzz, encarregada de atender às três metas definidas no principal objetivo desta dissertação. Delineando tais metas, esta ferramenta primeiramente assumiu a responsabilidade de arquiteturar interfaces gráficas interativas, capazes de mapear redes sociais pré-definidas.

A composição gráfica de redes insere-se necessariamente no campo de Visualização da Informação, e pode ser um fator preponderante para análises minuciosas de dados complexos, criando um canal facilitador de descobertas importantes.

Como segunda meta, o Fluzz viabilizará a criação de redes sociais através de simulações alicerçadas nos principais modelos de rede propostos na literatura, e em um novo modelo formalizado nesta dissertação. A utilização destas simulações mostra-se de fundamental importância para a compreensão da evolução das redes sociais, esclarecendo como diferentes variáveis inter-relacionadas influenciam neste processo.

Finalmente, a terceira meta da ferramenta refere-se à disponibilização de mecanismos de busca para a localização de pessoas específicas nas redes sociais, capazes de maximizar a probabilidade de influência entre os indivíduos. Este processo compreende a otimização de caminhos nas redes, que por sua vez, representam correntes formadas por um conjunto de indivíduos sequencialmente adjacentes por suas relações de amizade, principiado pelo indivíduo de origem e estendido até o indivíduo-alvo. Como as redes sociais violam a condição matemática da desigualdade triangular, contar com o prestígio dos amigos pelos caminhos, pode se tornar o diferencial para a criação de novas relações.

Além da influência exercida por um indivíduo perante as pessoas que o circundam indiretamente, cada elo intermediário pertencente a um caminho, pode contribuir significativamente para o reforço desta influência. Por isso, diversas análises sobre intensidade de influência em rede serão utilizadas para a criação de mecanismos inteligentes de buscas sociais, complementando, assim, o projeto da aplicação.

De acordo com os princípios da Engenharia de *Software*, o projeto de um sistema constitui a etapa encarregada de transformar os resultados da Análise de Requisitos em um conjunto de documentos e modelos capazes de serem interpretados durante o seu

desenvolvimento. Dessa forma, o projeto da aplicação Fluzz consistiu no levantamento de requisitos funcionais e não funcionais, na criação dos diagramas de casos de uso, na modelagem da estrutura de dados, e na definição de sua arquitetura.

9.1 Requisitos Funcionais

Os requisitos funcionais, além de descreverem as funcionalidades fornecidas, determinam os comportamentos esperados do sistema diante de diversas situações de entrada, viabilizando a identificação de ações a serem executadas em cada situação.

Perante o propósito da aplicação Fluzz, foram definidos 28 requisitos funcionais que abordam completamente as necessidades e características desejáveis da ferramenta.

9.1.1 Persistência dos dados gerados em um banco de dados

Este requisito visa permitir ao sistema armazenar em um banco de dados, as informações pessoais dos indivíduos das redes, além de suas relações de amizade. As informações pessoais devem contemplar: um identificador, o nome, e três atributos configuráveis que serão preenchidos conforme as necessidades particulares de cada rede.

9.1.2 Visualização bidimensional das redes por meio de grafos

O objetivo deste requisito é gerar o mapeamento gráfico em duas dimensões de redes sociais pré-estabelecidas, valendo-se para isto da construção de grafos representativos. Estes grafos serão constituídos por vértices que simbolizam as pessoas, e arestas que referenciarão os laços de amizades entre os indivíduos. Este é o principal requisito que atende à primeira meta da aplicação, sendo complementado por outros que também contribuem para os processos de mapeamento e transformação visual das redes.

9.1.3 Distinção visual dos vértices pela intensidade de cores

Distinguir visualmente os vértices das redes é o propósito deste requisito. A ideia por traz de tal recurso é gerar um mapeamento visual dos vértices através da sobreposição de cores, que viabilizem a diferenciação das pessoas de acordo com seu grau de conexão.

9.1.4 Seleção individual dos vértices e das arestas

Este requisito visa definir recursos que viabilizem a seleção individual de vértices ou arestas presentes nos grafos gerados. Esta seleção é pré-requisito para a realização de outras ações subsequentes, como a movimentação destes componentes.

9.1.5 Distinção de cores para vértices e arestas selecionados

Um recurso distintivo deve ser criado para facilitar a identificação visual por meio de cores, de vértices ou arestas selecionados no grafo.

9.1.6 Movimentação individual ou em grupo dos vértices

A aplicação deverá prover meios de movimentação dos vértices selecionados, podendo esta seleção contemplar um único, ou um conjunto dos mesmos.

9.1.7 Movimentação do grafo inteiro

Além da movimentação de vértices, o sistema deverá permitir a movimentação do grafo inteiro, viabilizando um melhor enquadramento de determinada perspectiva.

9.1.8 Variação da escala de visualização do grafo

Um recurso extremamente importante da ferramenta deverá ser provido através de mecanismos de *zoom*, permitindo que regiões específicas das redes sejam analisadas em diferentes escalas. O objetivo principal deste mecanismo é possibilitar a visualização de detalhes topológicos, muitas vezes despercebidos nos densos emaranhados das redes.

9.1.9 Rotação do grafo em torno de seus eixos

O sistema deverá permitir a rotação da imagem gerada pela aplicação, para que diferentes ângulos possam ser utilizados na visualização gráfica.

9.1.10 Deformação do grafo de maneira uniforme

Um recurso que também deve ser oferecido pela aplicação é a possibilidade de deformação da imagem. Esta ação visa criar diferentes perspectivas da mesma topologia.

9.1.11 Visualização do grafo por meio de diversos *layouts*

Diversos *layouts* de visualização de redes devem ser disponibilizados pelo sistema, para que os detalhes particulares de cada algoritmo de visualização, possam esclarecer informações muitas vezes confusas em determinado *layout*. Dessa forma, é importante que um mesmo grafo possa ser visualizado de diferentes formas, contribuindo com a descoberta de informações ocultas de uma topologia específica.

9.1.12 Cálculo automático dos pesos das arestas

Uma característica fundamental da aplicação é o cálculo do peso das conexões dos vértices, ou simplesmente, das arestas. Este cálculo deverá ser realizado automaticamente junto com o processo de visualização das redes, disponibilizando esta informação assim que o grafo for criado. No contexto da ferramenta, este peso refere-se à força do relacionamento das pessoas, materializado pelos laços de amizade. Esta medida é tida como de alta complexidade, por depender de variáveis intrínsecas a cada relacionamento.

Não obstante, de acordo com Granovetter [50], a força dos laços relacionais pode ser classificada basicamente em duas categorias: laços fortes e laços fracos. Cada uma destas subdivisões gera todo um universo interacional que atribui às amizades certo grau de intensidade, proximidade ou intimidade. Esta classificação conciliada à descoberta de Watts [107] sobre as dimensões sociais, possibilitou a criação de uma métrica para estimar o peso das conexões. Segundo Watts, as pessoas compartilham múltiplas dimensões sociais como religião, profissão, ou localidade, e quanto mais dimensões elas compartilharem, mais chances possuirão de se conhecer.

Utilizando este raciocínio, pode-se similarmente concluir que quanto mais dimensões duas pessoas compartilharem, mais intensa será a força da relação referente. A quantidade de amigos em comum, não pode determinar diretamente a quantidade de dimensões compartilhadas, porém é um indicador relevante para diferenciar a intensidade das relações. O motivo relaciona-se ao fato de que mais amigos em comum, aumentam significativamente as chances de diferentes dimensões serem incluídas neste processo.

Por isso, para o projeto da aplicação Fluzz, a força ou o peso das conexões será calculado dinamicamente através da quantidade de amigos em comum entre duas pessoas que se conhecem. Tal medida tem por objetivo diferenciar a força dos laços de amizade de determinada pessoa. Ela serve indiretamente como um bom referencial para aumentar as chances de influência mútua em rede, entendendo que quanto mais intensa for uma relação, maiores as chances de interação e cooperação recíproca.

9.1.13 Visualização dos pesos das arestas de forma opcional

A informação do peso das arestas deverá ser opcionalmente visualizada, através de algum mecanismo de apresentação deste dado na interface da aplicação.

9.1.14 Visualização do perfil de uma pessoa

O perfil de um indivíduo conterá todos os dados pessoais providos pela aplicação, além do cálculo do número de amigos e do coeficiente de aglomeração individual, podendo ser visualizado através da seleção de um vértice em particular.

9.1.15 Visualização do perfil de uma relação de amizade

Este perfil informará as duas pessoas que formam a amizade selecionada, bem como o peso referente à intensidade da relação, podendo ser visualizado através da seleção de uma aresta específica.

9.1.16 Apresentação da quantidade de vértices do grafo

O respectivo requisito visa totalizar a quantidade de vértices presentes no grafo, oferecendo a visualização de tal medida.

9.1.17 Apresentação da quantidade de arestas do grafo

A quantidade total de arestas do grafo também deve ser informada pela aplicação.

9.1.18 Apresentação do coeficiente de aglomeração da rede

Uma métrica importante que deve ser calculada é o coeficiente de aglomeração total da rede, disponibilizando-o assim que o grafo for gerado.

9.1.19 Apresentação da distância geodésica média da rede

O comprimento médio dos caminhos também deve ser calculado pela aplicação.

9.1.20 Geração de redes através dos modelos da literatura

Para este requisito alguns fatores devem ser considerados. Primeiro, deve ser possível gerar redes a partir de uma base inicial vazia. Segundo, o recurso de geração de redes também deve aceitar como origem alguma base de dados já pré-definida, e a partir de seus algoritmos, incrementar a quantidade de nós e conexões existentes. Terceiro, os modelos a serem utilizados neste processo deverão contemplar as redes aleatórias de Erdös e Rényi, as redes de mundo pequeno de Duncan Watts e Steven Strogatz, e as redes sem escala de Albert Barabási e Réka Albert, apresentadas no capítulo 2.

Finalmente, o critério para adição de novos vértices às redes deve seguir o seguinte esquema: a cada iteração um novo vértice será criado, e uma quantidade parametrizada de conexões para o mesmo deve ser efetivada. Dessa forma, com a dinâmica evolutiva da rede, todos os vértices a serem gerados devem se conectar à mesma taxa de conexão de seus pares anteriores. Consequentemente, não existirão vértices isolados, e dependendo do modelo de geração de redes utilizado, a distribuição de graus poderá ser mais ou menos uniforme. Este requisito, juntamente com o que será apresentado na próxima seção 9.1.21, atendem completamente à segunda meta da aplicação.

9.1.21 Geração de redes de acordo com um novo modelo

Um novo modelo de geração de redes também deverá ser disponibilizado pela aplicação, baseando-se em aperfeiçoamentos dos modelos já propostos na literatura. A ideia central deste novo modelo é que ele seja capaz de equilibrar as forças, de certa forma, conflitantes, dos dois principais modelos de geração de redes estudados pela Ciência das Redes: as redes de mundo pequeno e as redes sem escala.

Visando definir critérios que possam ser úteis para a análise da relevância desses modelos, quatro características observáveis nas redes por eles geradas foram mapeadas. O coeficiente de aglomeração, que informa a média de coesão do círculo de amizades na rede, a distância geodésica média, que mede o comprimento típico dos caminhos, o tamanho do maior *hub*, que avalia a existência de pessoas altamente conectadas, e a distribuição de graus da rede, que apresenta a probabilidade de um membro, aleatoriamente selecionado, ter um número específico de amigos.

Como já foi estudado, as redes geradas pelos modelos de mundo pequeno e livres de escala apresentam distinções importantes para cada uma dessas características avaliadas. A Tabela 9.1 apresenta, grosso modo, em uma escala de 0 a 5, as avaliações destes dois tipos de modelos perante tais características, juntamente com a avaliação do modelo aleatório que serve de parâmetro referencial.

Tabela 9.1: *Avaliação dos Modelos de Geração de Redes*

	CA	DGM	MH	DG
Redes Aleatórias	0	4	2	Normal
Redes de Mundo Pequeno	5	2	1	Normal
Redes Sem Escala	1	5	5	Leis de Potência

CA=Coeficiente de Aglomeração, DGM=Distância Geodésica Média, MH=Maior Hub, e DG=Distribuição de Graus.

É importante esclarecer que tais avaliações não servem como métricas isoladas, devendo ser utilizadas apenas como parâmetro comparativo entre os diferentes modelos. Uma rede do tipo livre de escala não pode ser caracterizada como fracamente aglomerada. Não obstante, se estas redes forem comparadas com redes de mundo pequeno, certamente apresentarão aglomerações bem menores. Por isso nesta característica, cada rede recebeu respectivamente as avaliações 1 e 5.

Isso ocorre basicamente pelo fato das redes sem escala não utilizarem a estrutura social adjacente como um fator importante no processo de conexão. A única informação analisada em seu processo conectivo é o grau de conexão de cada vértice, tornando algumas pessoas muito mais prováveis de receberem conexões do que outras. Na sociedade, como bem identificou Watts, a estrutura social de uma pessoa torna alguns indivíduos mais próximos do que outros, independente do grau de conexão.

Em contrapartida, outro processo parece apontar um ponto fraco nas redes de mundo pequeno. Isoladamente estas redes não podem ser rotuladas como redes que apresentam caminhos típicos longos. Todavia, ao se compará-las com as redes livres de escala, o comprimento típico dos caminhos mostra-se bem maior.

A explicação deste processo foi realizada por Barabási. Como Watts assumiu que a distribuição de graus das redes, basicamente obedecia a uma distribuição normal, ele não foi capaz de identificar os *hubs*. Estes indivíduos modificam a distribuição de graus da rede, encurtando distâncias à medida que se tornam mais bem conectados.

Dessa forma, o equilíbrio de forças objetivado pelo novo modelo a ser proposto, deve ser capaz de uniformizar positivamente as quatro características mensuradas, aproximando-se da real topologia da sociedade. Atingido este propósito, o modelo pode representar todo um novo universo geracional de redes, estabilizando pontos fracos, e permitindo que as principais descobertas dos modelos anteriores possam ser conciliadas.

Denominado como *Modelo Gama*, este novo modelo de geração de redes basicamente posiciona os padrões de Watts e Strogatz, e Barabási e Albert sob a mesma égide. Um parâmetro ajustável de controle *gama* foi definido como a variável de interpolação entre estes dois modelos, criando toda uma família de regras de interação, cada qual especificada por um valor particular de γ.

Na prática, o que é feito é dar um valor, entre zero e um, ao parâmetro ajustável, determinando homogeneamente a forma de conexão dos vértices adicionados às redes. Assim, se *gama* for zero, os grafos apresentarão topologias de redes de mundo pequeno. No outro extremo, quando *gama* é igual a um, redes livres de escala caracterizarão a estrutura dos grafos. A Figura 9.1 apresenta a dinâmica evolutiva deste modelo proposto.

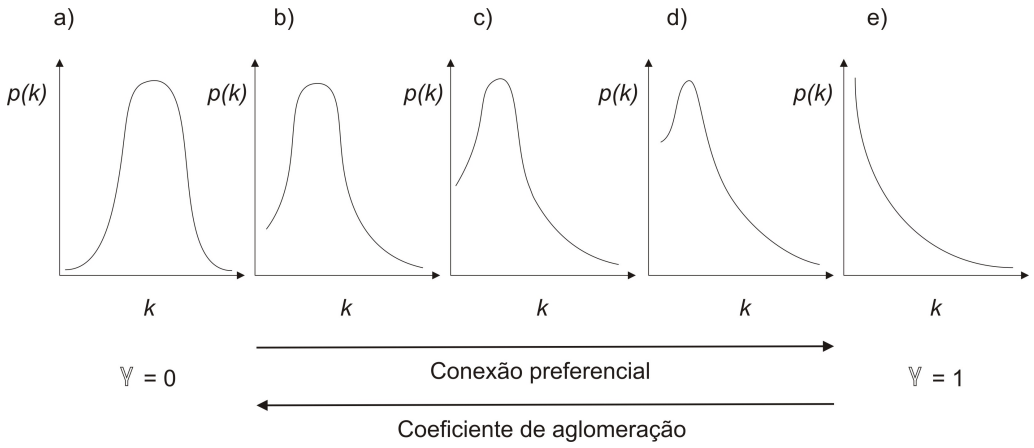


Figura 9.1: *Modelo Gama.*

A indefinição é exatamente o que acontece no meio. Quando o valor de *gama* é pequeno, a estrutura social local torna-se altamente importante para a determinação dos *links*, porém algumas conexões preferenciais certamente são realizadas. Não obstante,

quando o valor do parâmetro é alto, conexões preferenciais dominam a intensão conectiva dos vértices, todavia algumas ligações locais podem ser realizadas.

A expectativa é que em alguma região intermediária, não muito próxima dos extremos, algo surpreendente possa ser descoberto. É esta revelação que determinará o melhor ponto de equilíbrio dos dois modelos base, estabelecido pelo *Modelo Gama*.

9.1.22 Importação dos dados de redes pré-definidas

Quando a necessidade não for a geração automática de redes, e sim a análise de redes sociais pré-existentes, o usuário pode se valer de aplicações externas para efetuar a importação destes dados. Desta forma, será possível utilizar os recursos da aplicação para realizar análises em redes sociais reais.

9.1.23 Adição de conexões às redes existentes

A aplicação deverá prover recursos para a adição de novas conexões aos vértices existentes nas redes, de acordo com algum dos modelos implementados.

9.1.24 Detecção de caminhos entre duas pessoas

Um dos principais requisitos da aplicação é a capacidade de pesquisa de caminhos entre dois indivíduos nas redes, maximizando a probabilidade de influência entre os mesmos. Para a realização deste procedimento, duas características são essenciais: a minimização de distância, e a maximização dos pesos referentes às relações de amizade.

Entende-se por minimização de distância, a possibilidade de comparar diversos caminhos, e decidir pelo que apresenta menor custo, que em redes sociais, refere-se ao grau de separação existente entre dois indivíduos. De acordo com as pesquisas de Christakis e Fowler [30], quanto mais distantes estão duas pessoas na rede, menores as chances de influência, ou cooperação recíproca.

Além disso, os pesquisadores identificaram na sociedade a regra dos três graus de influência, que torna a fronteira dos três graus de separação, um provável limite para a influência humana. Para distâncias maiores, tal influência deixa de ter impacto perceptível, e um possível caminho criado entre duas pessoas torna-se inutilizável.

Sendo assim, o processo de minimização de distância deve se concentrar em menores graus de separação, priorizando essencialmente a região delimitada pelos primeiros três graus, criando um caminho útil que a pessoa de origem pode percorrer. Correntes de amizade como estas podem representar a grande diferença entre uma tentativa de contato direto, ou intermediada de certa forma pelas relações sociais referenciais. Estas referências, que de fato, representam os indivíduos de um caminho, podem ser utilizadas de

diferentes maneiras pelas pessoas interessadas, na tentativa de criar um canal social necessário para o estabelecimento ou a efetivação da conexão desejada.

As consequências dessa integração criada pelos seres humanos produzem efeitos de rede dos mais notáveis, dentre eles a produção do capital social, que representa a capacidade das pessoas de trabalharem juntas de forma fácil e eficiente, com base na confiança. Se as ligações de longa distância mantêm as redes conectadas, o capital social, criado pela aglomeração, proporciona numerosos elos fortes, e um contexto de indivíduos firmemente engastados, o que viabiliza a criação de correntes sociais cooperativas.

Como comprovação desse fenômeno, pesquisadores da Universidade de Chigago apresentaram os resultados de um estudo sobre 2.777 pessoas nos Estados Unidos, oferecendo uma descrição do comportamento social relacional. A Tabela 9.2 mostra quem apresentou casais em diferentes tipos de relacionamento [65].

Tabela 9.2: *Apresentações em Sociedade*

	Relacionamentos próximos		Relacionamentos distantes					
Tipo de Relacionamento	Família	Amigo	Trabalho	Escola	Vizinho	Autoapres.	Outra	Participantes
Casamentos	15%	35%	6%	6%	1%	32%	2%	1.287
Coabitação	12%	40%	4%	1%	1%	36%	3%	319
Parcerias	8%	36%	6%	4%	1%	42%	1%	920
Parceiras de curto prazo	3%	37%	3%	4%	2%	47%	2%	251

Cerca de 61% das pessoas no estudo conheceram seus parceiros após terem sido introduzidas por alguém que conheciam, enquanto apenas 39% se conheceram via autoapresentação. Dessa forma, embora exista a possibilidade de encontros entre estranhos, a maioria das pessoas encontrou parceiros ao conhecer os amigos dos amigos e outras pessoas às quais elas estavam indiretamente conectadas.

Outra pesquisa com mais de dois milhões de citações entre patentes identificou o efeito das redes sociais na difusão das ideias entre inventores [91]. A pesquisa mostrou que havia forte probabilidade dos inventores com laço de colaboração direto citarem um ao outro, quatro vezes mais do que seria esperado devido ao acaso. O efeito se estendeu ainda mais pela rede, sendo que em dois graus de separação, eles teriam aproximadamente 3,2 vezes mais probabilidade de citar um ao outro e, em três graus teriam 2,7 vezes mais probabilidade. Além de três graus, o efeito praticamente desaparece.

Obviamente, as pessoas confiam muito nos amigos e na família para todo tipo de relacionamento, já que a apresentação socialmente intermediada torna-se menos arriscada e mais informativa. Granovetter constatou que se alguém souber que um amigo de um amigo se encontrou com determinada pessoa, há uma espécie de corrente entre eles. Isto torna as intenções das pessoas menos suspeitas, uma vez que elas "exercem influência através da corrente de maneira a restringir posturas interesseiras" [51].

Por essa perspectiva, um segundo procedimento referente à maximização da probabilidade de influência deve ser incluído no processo de busca: maximizar o peso das conexões. Como estes pesos representam a intensidade da relação entre dois indivíduos,

quanto maior o peso, maior será a chance de influência mútua. Portanto, o processo de pesquisa de caminhos deve contemplar ambos os procedimentos para efetivar-se como um recurso relevante: minimizar custos de distância e maximizar o peso das relações.

Para a realização dessa pesquisa, duas possibilidades devem ser disponibilizadas. Na primeira, o usuário poderá informar o código de identificação dos vértices de origem e destino, e na segunda, a identificação do vértice de destino poderá ser realizada através da informação de seu nome. Ao término do processamento da pesquisa, a aplicação deverá apresentar visualmente e descritivamente o melhor caminho encontrado no grafo. Este requisito, juntamente com o que será apresentado na seção posterior 9.1.25, atendem completamente à terceira meta da aplicação Fluzz.

9.1.25 Detecção de caminhos entre uma pessoa e várias outras

O recurso provido pelo requisito apresentado na seção 9.1.24, também deverá ser estendido para múltiplos vértices de destino. Neste caso, o melhor caminho será aquele dentre todos os melhores caminhos encontrados para cada um destes vértices.

Essa situação, só será possível caso o usuário não utilize na pesquisa, o código de identificação do vértice de destino. Neste caso, ele pode usufruir dos três campos configuráveis, ou do próprio nome da pessoa, para alimentar os atributos dos possíveis indivíduos-alvo na rede. Consequentemente, a aplicação realizará filtros para selecionar os indivíduos possuidores destes atributos, assinalando-os como alvos para a pesquisa.

9.1.26 Criação de filtros para visualização dos caminhos

Alguns filtros devem ser oferecidos pela aplicação, no que tange à visualização dos caminhos encontrados. Um primeiro filtro deve apresentar somente o caminho encontrado, ocultando os demais vértices e arestas não pertencentes ao mesmo. Uma segunda opção, além do disponibilizado no primeiro filtro, deverá apresentar todos os vértices que possuem conexões com algum dos vértices pertencentes ao respectivo caminho. Por fim, uma terceira opção deverá apresentar, além do próprio caminho encontrado, os vértices adicionais que compartilham conexões com mais de um vértice deste caminho.

A intensão dos filtros é minimizar recursos computacionais gráficos, além de possibilitar a visualização de indivíduos pertencentes ao círculo social dos elementos que compõem o melhor caminho identificado.

9.1.27 Salvamento da imagem da rede gerada

O sistema deverá prover a opção de salvamento da imagem do grafo gerado, criando um arquivo com extensão *jpg* que poderá ser utilizado para análises futuras.

9.1.28 Deleção da base de dados

É importante que a aplicação possua um meio de apagar todos os dados criados, devolvendo ao sistema seu estado inicial.

9.2 Requisitos Não Funcionais

Os requisitos não funcionais definem os atributos de qualidade do sistema ao executar suas funcionalidades. Estes requisitos não especificam funções específicas, aplicando-se ao sistema como um todo, e caracterizando-se por restrições sobre os serviços fornecidos pelo mesmo.

Para o projeto da aplicação Fluzz, foram definidos 8 requisitos não funcionais, que visam garantir características mínimas de qualidade para a aplicação gerada.

- **Modularidade:** A modularização da ferramenta deve ser alcançada com o projeto e a utilização de agentes de *software*;
- **Simplicidade:** Novos agentes poderão ser agregados à ferramenta de forma simples e pouco custosa, acrescentando recursos adicionais à aplicação;
- **Paralelismo:** O processamento paralelo dos agentes deverá ser provido pelo sistema;
- **Eficiência:** O sistema deverá processar e interpretar corretamente todos os modelos definidos para se atingir os resultados esperados;
- **Confiabilidade:** Mesmo diante de circunstâncias hostis ou inesperadas, a aplicação deverá realizar e manter seu funcionamento de circunstâncias normais;
- **Portabilidade:** A ferramenta poderá ser executada em diversas plataformas;
- **Manutenibilidade:** As ações de manutenção no sistema deverão ser viabilizadas de modo preciso e seguro, através da implementação de padrões de projeto;
- **Usabilidade:** O *software* deve possuir interface instintiva para facilitar o uso de seus recursos.

9.3 Diagramas de Caso de Uso

Um caso de uso é uma unidade funcional provida pelo sistema. Cada caso de uso neste projeto relacionará os diversos requisitos funcionais de determinada classe, que contempla um conjunto de ações semelhantes. Dois atores serão definidos, o usuário e uma possível aplicação externa que interagem com o sistema. Ambos serão responsáveis por operacionalizar todos os 9 casos de uso definidos para os 28 requisitos funcionais.

A vinculação de cada requisito funcional ao seu respectivo caso de uso foi realizada de acordo com a Tabela 9.3.

Tabela 9.3: *Mapeamento de Requisitos Funcionais e Casos de Uso*

Requisito Funcional	Caso de Uso
Persistência dos dados gerados em um banco de dados	Gravar rede
Visualização bidimensional das redes por meio de grafos Distinção visual dos vértices pela intensidade de cores Visualização do grafo por meio de diversos *layouts* Visualização dos pesos das arestas de forma opcional Visualização do perfil de uma pessoa Visualização do perfil de uma relação de amizade	Visualizar grafo
Movimentação individual ou em grupo dos vértices Movimentação do grafo inteiro Variação da escala de visualização do grafo Rotação do grafo em torno de seus eixos Deformação do grafo de maneira uniforme Seleção individual dos vértices e das arestas Distinção de cores para vértices e arestas selecionados	Interagir com o grafo
Apresentação da quantidade de vértices do grafo Apresentação da quantidade de arestas do grafo Apresentação do coeficiente de aglomeração da rede Apresentação da distância geodésica média da rede Cálculo automático dos pesos das arestas	Apresentar métricas
Geração de redes através dos modelos da literatura Geração de redes de acordo com um novo modelo Importação dos dados de redes pré-definidas Adição de conexões às redes existentes	Gerar redes
Detecção de caminhos entre duas pessoas Detecção de caminhos entre uma pessoa e várias outras	Localizar caminhos
Criação de filtros para visualização dos caminhos	Criar filtros
Salvamento da imagem da rede gerada	Salvar imagem
Deleção da base de dados	Deletar rede

A Figura 9.2 apresenta o diagrama de casos de uso da aplicação Fluzz.

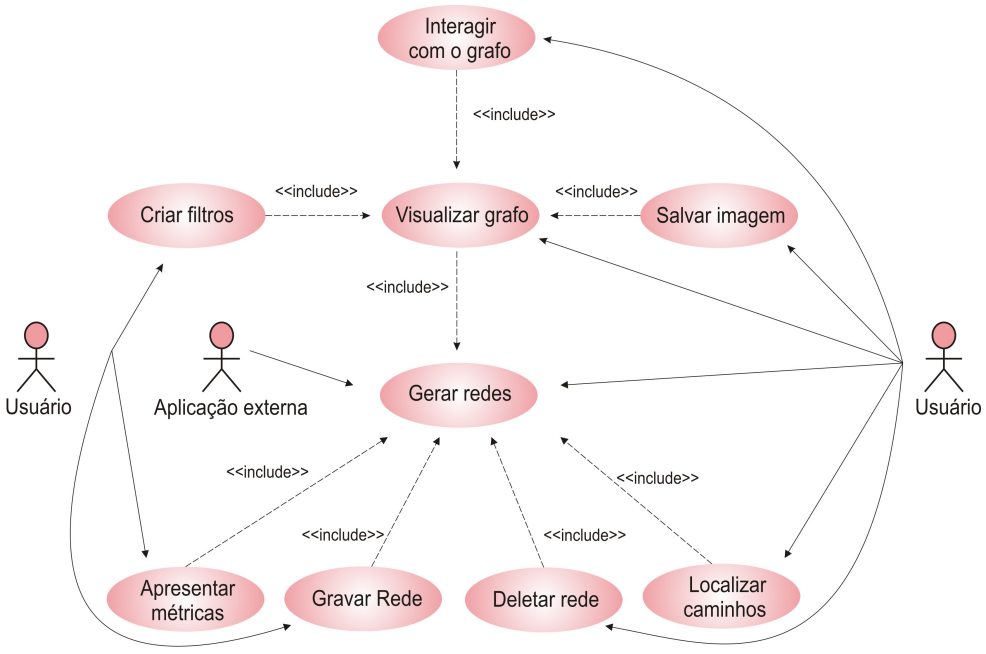


Figura 9.2: *Diagrama de Casos de Uso.*

9.4 Modelagem da Estrutura de Dados da Aplicação

O processo de modelagem da estrutura de dados da aplicação visa definir o arcabouço necessário para que as informações geradas pelo *software* possam ser persistidas corretamente em um banco de dados relacional.

Para a realização dessa modelagem foram criados o Modelo Conceitual e o Modelo Relacional com seus respectivos Dicionários de Dados, que devem ser seguidos durante as fases de desenvolvimento e implantação do sistema.

A Figura 9.3 apresenta o Modelo Conceitual criado, e a Tabela 9.4 seu respectivo Dicionário de Dados.

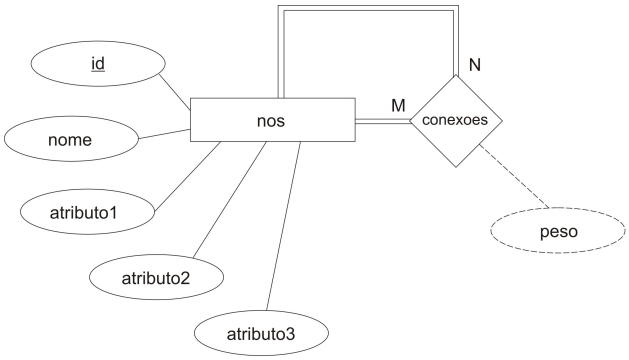


Figura 9.3: *Modelo Conceitual da Aplicação Fluzz.*

Tabela 9.4: *Dicionário de Dados do Modelo Conceitual*

Entidade	*nos*			
Descrição	Armazena os dados das pessoas			
Atributos	**Nome**	**Tipo**	**Descrição**	**Identificador**
	id	Número	Código de Identificação das pessoas	Sim
	nome	Texto	Nome da pessoa	Não
	atributo1	Texto	1^o atributo configurável	Não
	atributo2	Texto	2^o atributo configurável	Não
	atributo3	Texto	3^o atributo configurável	Não
Relacionamento	*conexoes*			
Descrição	Armazena as relações de amizade das pessoas			
Atributos	**Nome**	**Tipo**	**Descrição**	**Identificador**
	peso	Número	Quantidade de amigos em comum	Não
Cardinalidade	*nos(1:M) X nos(1:N)*			

Neste modelo, o atributo *peso* pertencente ao relacionamento *conexoes*, foi definido como derivado, pois se trata de uma informação calculada pelo sistema.

Por conseguinte, foi criado o Modelo Relacional visualizado na Figura 9.4, bem como seu respectivo Dicionário de Dados presente na Tabela 9.5.

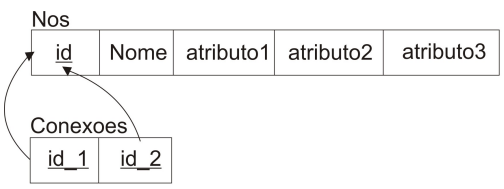


Figura 9.4: *Modelo Relacional da Aplicação Fluzz.*

Tabela 9.5: *Dicionário de Dados do Modelo Relacional*

Relação	*nos*						
Descrição	Idem Tabela 9.4						
Atributos	**Nome**	**Tipo**	**Descrição**	**Único**	**Nulo**	**Chave**	**Observação**
	id	Número(8)	Idem Tabela 9.4	Sim	Não	PK	
	nome	Texto(100)	Idem Tabela 9.4	Não	Não		
	atributo1	Texto(100)	Idem Tabela 9.4	Não	Sim		
	atributo2	Texto(100)	Idem Tabela 9.4	Não	Sim		
	atributo3	Texto(100)	Idem Tabela 9.4	Não	Sim		
Relação	*conexoes*						
Descrição	Idem Tabela 9.4						
Atributos	**Nome**	**Tipo**	**Descrição**	**Único**	**Nulo**	**Chave**	**Observação**
	id_1	Número(8)	Código de Identificação das pessoas	Sim	Não	PK	FK da relação *nos*
	id_2	Número(8)	Código de Identificação das pessoas	Sim	Não	PK	FK da relação *nos*

9.5 Arquitetura da Aplicação

Esta seção dedica-se à apresentação da arquitetura da aplicação Fluzz, detalhando seus três componentes principais: o agentePrincipal, a Sociedade de Agentes de Pesquisa, e o Repositório de Dados, como ilustrado na Figura 9.5.

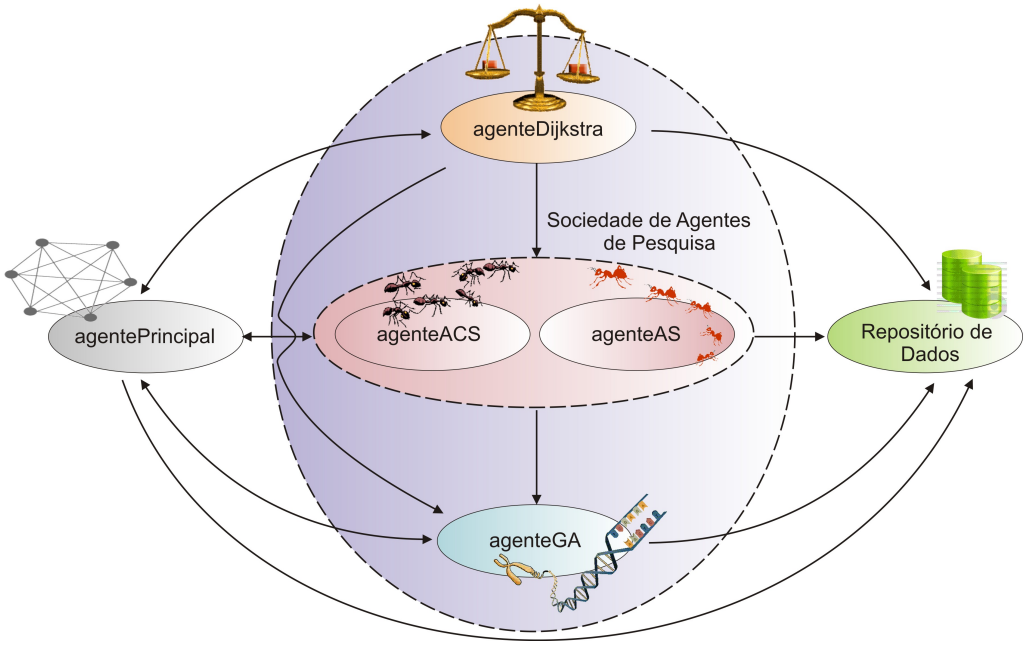


Figura 9.5: *Arquitetura da Aplicação Fluzz.*

9.5.1 O agentePrincipal

O agentePrincipal será o componente fundamental da aplicação, sendo responsável individualmente por realizar 8 dos 9 casos de uso estipulados. Somente o caso de uso *Localizar caminhos* não será de sua inteira responsabilidade. Não obstante, é este agente que iniciará o processo de busca propriamente dito, através do envio de uma mensagem para o primeiro agente da Sociedade de Agentes de Pesquisa.

A partir desse momento, uma série de comunicações será realizada pelos demais elementos da Sociedade, que de forma cooperativa devem buscar pela melhor solução do problema. Quando um dos agentes encontrar a solução procurada, de acordo os critérios pré-estabelecidos nos modelos definidos, uma mensagem deverá ser retornada para o agentePrincipal. Consequentemente, este agente extinguirá os outros agentes ainda ativos no processo de busca, e representará graficamente o caminho encontrado.

Dessa forma, o agentePrincipal possuirá um papel fundamental no procedimento de busca nas redes, sendo responsável pela inicialização e encerramento deste processo.

9.5.2 A Sociedade de Agentes de Pesquisa

A Sociedade de Agentes de Pesquisa contemplará quatro agentes que trabalharão cooperativamente no processo de busca em redes. O agenteDijkstra será o primeiro elemento a ser ativado na Sociedade. Sua denominação é motivada devido ao algoritmo utilizado por este agente, o Algoritmo de Dijkstra [32].

Os próximos elementos a serem ativados na Sociedade são os agentes ACS e AS, que iniciam o processo de busca paralelamente ao seu comparsa inicial. As abordagens utilizadas por estes dois últimos agentes seguem respectivamente as metaheurísticas de Otimização por Colônia de Formigas *Ant Colony System* e *Ant System* [40].

Ainda existe a possibilidade do agenteGA ser ativado por algum desses agentes de segundo nível. Se esta ação ocorrer, o agenteGA que implementa um Algoritmo Genético [64], também passará a cooperar com os outros elementos da Sociedade.

Para o projeto da aplicação Fluzz, foi definido que o agenteDijkstra deverá ser o único responsável por devolver caminhos de até três graus de separação. O motivo consiste no fato de que a pesquisa realizada por este agente é capaz de retornar, caso exista, uma solução ótima do problema, devido a sua característica exploratória.

Como a quantidade de conexões de cada pessoa em sociedade é um número relativamente pequeno, e mesmo contando com um crescimento exponencial deste número à medida que se afasta do ponto de origem, até três graus de separação, o montante resultante dos caminhos ainda pode ser analisado rapidamente por este agente.

Consequentemente, dentro desse limite, o melhor entre todos os caminhos existentes para determinada pesquisa pode ser encontrado de modo eficiente. É exatamente esta condição que torna o agenteDijkstra extremamente importante para a aplicação, que é capaz de localizar o melhor caminho pertencente à região factível de influência humana, contribuindo significativamente para a localização de prováveis parceiros em sociedade.

A fim de definir uma forma para a análise qualitativa dos caminhos, o modelo *BestWay* foi proposto para integrar a lógica de avaliação do agenteDijkstra, tendo como alicerce um tripé avaliativo que diferencia caminhos seguindo o seguinte critério:

1. **Análise da quantidade de laços fracos:** O primeiro elemento a ser verificado pelo modelo é a quantidade total de laços fracos identificados dentro da zona de influência humana, referente aos três graus de separação. Esta medida possui grau de dominância perante as outras duas formas de análise, o que significa ser capaz de determinar isoladamente a solução, caso seus valores nos caminhos comparados sejam diferentes. Apesar da importância deste tipo de laço para formação de redes conexas, o objetivo deste elemento foi de minimizar relações de fraca intensidade nos caminhos encontrados, que aumentam a probabilidade de quebra da corrente cooperativa. Como a determinação do que pode representar um laço fraco é subjetiva, pois depende das características conectivas das pessoas, a aplicação deve possuir recursos parametrizáveis para esta medida, sendo o usuário responsável por informar um valor que definirá a fronteira entre as forças dos laços de amizade: fortes ou fracos. Como exemplificação, a Figura 9.6 ilustra duas situações, em que o primeiro elemento do modelo definiu qual é o melhor caminho. Na primeira imagem à esquerda da figura, três caminhos separam os vértices 1 e 5. Supondo que o usuário definiu na aplicação, que um laço fraco corresponde às ligações com pesos menores que 5, o caminho (1,5) possuirá um laço fraco, o caminho (1,4,2,5) possuirá dois, e o caminho (1,3,5) possuirá nenhum. Como todos estes caminhos estão na zona de influência humana, o agente determina que o caminho (1,3,5) é a melhor escolha por possui a menor quantidade de laços fracos, o que provoca a violação da desigualdade triangular. Na segunda imagem à direita da figura, dois caminhos separam os vértices 1 e 3. Com a mesma definição de laço fraco feita anteriormente, o caminho (1,4,2,5,3) não possui laços fracos, enquanto que o caminho (1,4,3) possui um. Não obstante, o caminho (1,4,2,5,3) possui mais de três graus de separação, o que não ocorre com o caminho (1,4,3), que será então o escolhido pelo agente.

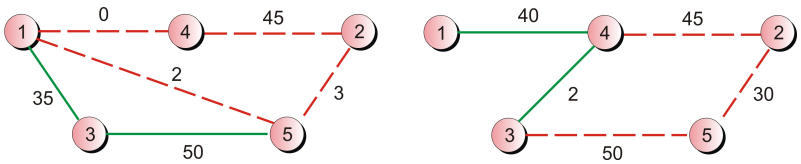


Figura 9.6: *Primeiro Elemento do Modelo BestWay.*

2. **Análise do grau de separação:** Quando o agenteDijkstra não for capaz de distinguir entre dois caminhos através do primeiro elemento do modelo, um segundo critério de avaliação qualitativa entra em cena, desta vez referindo-se à distância através da rede entre as pessoas. Para exemplificar o processo de análise do segundo elemento, a Figura 9.7 ilustra duas situações diferentes. A imagem à esquerda da figura apresenta dois caminhos que separam os vértices 1 e 5. Utilizando-se da análise do primeiro elemento do modelo, e com a mesma definição de laço fraco feita na exemplificação anterior, verifica-se que ambos os caminhos (1,4,2,5) e (1,3,5)

possuem um laço fraco, e estão dentro da zona de influência humana. Esta situação obriga o agente a partir para a análise do próximo elemento, que constata que o caminho (1,3,5) é menor, e portanto deverá ser escolhido. Na imagem à direita da figura, ambos os caminhos que separam os vértices 1 e 6, (1,4,2,5,6) e (1,4,3,2,5,6) estão fora da zona de influência, e por isso serão diferenciados somente pelo grau de separação. Isto faz com que o caminho (1,4,2,5,6) seja escolhido por ser menor que o caminho (1,4,3,2,5,6), mesmo possuindo um laço fraco a mais.

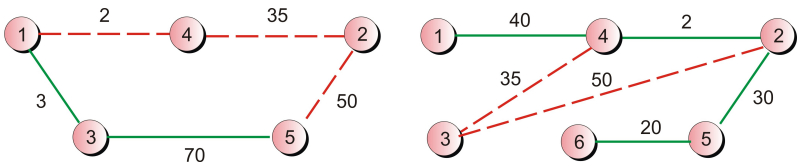


Figura 9.7: *Segundo Elemento do Modelo BestWay.*

Em um processo similar ao do primeiro elemento do modelo, a característica referente ao grau de separação domina o último critério de avaliação utilizado.

3. **Análise do valor acumulado dos pesos:** Se ainda houver uma situação em que dois caminhos diferentes apresentem a mesma quantidade de laços fracos e o mesmo grau de separação, o terceiro e último elemento avaliativo do modelo deve ser utilizado. Trata-se neste ponto do valor do peso acumulado pelo caminho, que quanto maior for, melhor qualificará o percurso. O objetivo é selecionar laços de amizade mais intensos, que provavelmente estarão mais aptos a cooperarem mutuamente. Este processo é ilustrado na Figura 9.8, onde existem três melhores caminhos de comprimento 2, que ligam os vértices 1 e 5. Ainda com a mesma definição de laço fraco feita anteriormente, nenhum dos caminhos apresenta laços fracos, sendo que o caminho (1,3,5) totaliza um peso 70, o caminho (1,4,5) totaliza um peso 65, e o caminho (1,2,5) totaliza um peso 75. A classificação em ordem decrescente de qualidade para este exemplo seria: (1,2,5), (1,3,5), e (1,4,5).

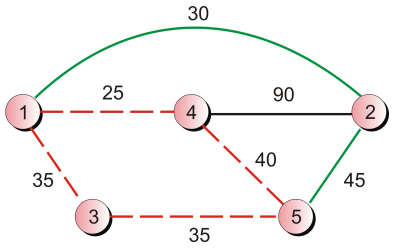


Figura 9.8: *Terceiro Elemento do Modelo BestWay.*

A Figura 9.9 ilustra o modelo *BestWay* seguido pelo agenteDijkstra.

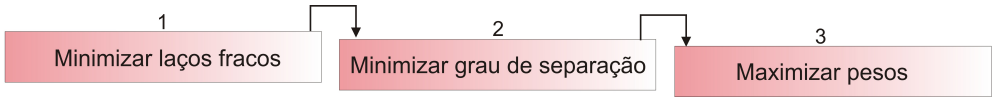


Figura 9.9: *Representação do Modelo BestWay.*

Como já foi estudado, a partir do quarto grau de separação, a possibilidade de influência recíproca em sociedade torna-se desprezível. Portanto, encontrar uma pessoa a quatro graus de distância ou mais, muitas vezes não terá um significado prático, pelo menos, não à primeira vista. Todavia, um caminho de qualquer tamanho sempre representará uma pista, mesmo que esta seja realmente pequena.

Em outras palavras, um caminho longo não deve ser de utilidade imediata, pois a fronteira dos três graus de separação parece ser o limite da violação da desigualdade triangular. A partir do próximo ponto da escala de distância, a influência não é mais um fator a ser considerado no processo conectivo. Em termos práticos, é melhor optar pela autoapresentação, e tentar um contato direto ao invés de intermediá-lo.

De qualquer forma, o caminho longo ainda existirá, e uma única conexão criada por algum dos indivíduos intermediários deste caminho, pode empurrar a distância entre os vértices de origem e destino, para dentro da zona factível de influência. Além disso, conhecer quem são os indivíduos de uma corrente de amizade, muitas vezes viabiliza o reconhecimento de pessoas próximas na vida real, que estão momentaneamente distantes na vida *online*. Consequentemente, conexões estratégicas podem ser realizadas pelo próprio indivíduo de origem, encurtando a distância final para o indivíduo de destino.

É por esta razão que caminhos aparentemente longos não devem ser desprezados por completo, pois eles podem prover informações importantes sobre o interior das redes.

Não obstante, um problema emerge quando se deve pesquisar além da fronteira da influência: a complexidade de ramificação das redes. A quantidade de laços a partir desta região, de acordo com o tamanho da rede, pode atingir a casa de 8 dígitos ou mais, representando um vasto ambiente de pesquisa. Sendo assim, o agenteDijkstra poderá ter sua performance prejudicada, dependendo da localização dos indivíduos-alvo nas redes.

De acordo com o projeto da aplicação, assim que uma busca for efetivada pelo usuário, o agentePrincipal deverá ativar em sequência os seguintes agentes: agenteDijkstra, agenteACS e agenteAS. Todos eles iniciarão suas atividades de pesquisa simultaneamente, cada um seguindo suas respectivas lógicas ou heurísticas particulares. No entanto, o agenteACS e o agenteAS só poderão contribuir com suas informações, quando o agenteDijkstra superar a fronteira dos três graus de separação.

Nesse ponto, caso, por exemplo, o agenteACS tenha encontrado uma solução com quatro graus de distância, antes do agenteDijkstra, imediatamente ele pode enviar uma mensagem ao agentePrincipal informando o ocorrido. O agentePrincipal por sua vez, extinguirá todos os agentes ativos, e devolverá a solução encontrada pelo agenteACS ao usuário. O mesmo processo vale também para o agenteAS.

Os agentes baseados no comportamento das formigas são excelentes metaheurísticas aplicáveis a problemas de otimização. Graças às suas orientações heurísticas, e à retroalimentação realizada pelo depósito de feromônio, as formigas artificiais são capazes

de navegar por grandes redes, e encontrar caminhos curtos em um tempo reduzido.

Como a região, além dos três graus de distância, não é caracterizada a priori pela existência de caminhos úteis aos seres humanos, não existe a necessidade desses agentes localizarem caminhos ótimos pela rede. Por isso, a utilização de metaheurísticas de otimização encaixa-se perfeitamente neste novo contexto.

Nenhum apurado modelo avaliativo de caminhos será então utilizado pelas formigas artificiais, que basicamente se orientarão somente pela análise do grau de separação, o que contribui para o aumento da performance do processo de busca. Caminhos encontrados com menos de quatro graus de distância deverão ser desconsiderados, uma vez que compete somente ao agenteDijkstra obter resultados nesta região.

Finalmente, o último agente da Sociedade, o agenteGA, pode ser ativado por um desses dois últimos agentes metaheurísticos descritos, baseados em colônias de formigas. O agenteGA utiliza como metaheurística os Algoritmos Genéticos, com uma fundamental definição para a composição de sua população inicial.

Para evitar problemas com a criação de caminhos inviáveis, todos os cromossomos da população do agenteGA deverão ser originalmente formigas artificiais, oriundas do agenteACS ou do agenteAS. Assim que as duas primeiras formigas são formadas por um destes agentes, imediatamente serão codificadas como cromossomos, e o processo de melhoria genética se desenvolverá de acordo com as operações propostas na literatura.

O objetivo do agenteGA é produzir uma diversificação dos caminhos gerados, uma vez que as formigas artificiais podem convergir para algum ótimo local. Consequentemente, caso este agente, que também trabalha além dos três graus de separação, encontre uma solução antes de seus comparsas, o processo se encerrará de forma análoga ao descrito anteriormente para o agenteACS e para o agenteAS. Além disso, o mesmo processo para avaliação de caminhos destes últimos agentes também será utilizado pelo agenteGA.

Sempre que o agenteDijkstra adentrar um novo nível de distância na pesquisa, uma mensagem deverá ser enviada pelo mesmo, a todos os outros agentes ativos na ferramenta. A intensão é que os agentes metaheurísticos possam estar conscientes da distância requerida a cada momento, para que um determinado caminho obtido possa ser considerado como solução do problema. Todos os agentes da Sociedade podem acessar o Repositório de Dados da aplicação, a fim de navegar pela rede à procura de soluções.

9.5.3 O Repositório de Dados

O Repositório de Dados da aplicação armazena todos os dados das redes geradas. Assim que a base de dados da ferramenta é alimentada, seja por uma aplicação externa, ou através do caso de uso *Gerar redes*, este repositório é responsável por manter os dados íntegros e disponíveis para análises ou utilizações futuras.

CAPÍTULO 10

Desenvolvimento da Aplicação

Após a concepção do projeto da aplicação Fluzz, foi iniciada a última etapa de sua criação: a codificação. Este desenvolvimento alicerçou-se inteiramente nos requisitos, na modelagem e na arquitetura outrora definidos, viabilizando a produção de um *software* plenamente de acordo com seus objetivos.

Os estudos teóricos realizados nesta dissertação foram capitais para que técnicas de programação pudessem ser eficientemente aplicadas na codificação dos algoritmos projetados, o que permitiu a verificação e a validação dos modelos propostos. Desta forma, este capítulo descreve como as tecnologias escolhidas para a criação do Fluzz foram empregadas, além de apresentar as técnicas utilizadas na implementação dos requisitos.

Visando atender aos objetivos deste capítulo de maneira esclarecedora, a descrição das funcionalidades das principais classes criadas, além da apresentação gráfica dos resultados obtidos pela codificação dos requisitos, serão introduzidas à medida que estes últimos recebam o foco da análise, contextualizando-os às suas demandas originais.

10.1 Tecnologias Empregadas

Diversas tecnologias foram utilizadas para a implementação do Fluzz, tendo cada uma contribuído para a produção final do sistema. A seguir, uma breve análise das mesmas é realizada, contemplando a apresentação de seus principais recursos e finalidades.

10.1.1 Java 7

Plataforma Java é a denominação dada ao ambiente computacional criado pela empresa Sun Microsystems e hoje pertencente à Oracle. O Java SE ou *Java Platform Standard Edition* é a tecnologia base da plataforma, incluindo o ambiente de execução e as bibliotecas comuns, através da qual a aplicação Fluzz foi inteiramente desenvolvida.

A edição do Java voltada para a criação de aplicações corporativas e para Internet, o Java EE ou *Java Platform Enterprise Edition*, foi utilizada para a execução do Fluzz na Web através de *Applets*, ou *softwares* aplicativos executados em um *browser*.

A plataforma Java permite desenvolver aplicativos utilizando qualquer linguagem criada para seu contexto, sendo a linguagem padrão sua homônima: a linguagem Java. Portanto, Java também firma-se como uma linguagem de programação de código fonte aberto, que diferentemente das linguagens convencionais compiladas para código nativo, executa compilações para *bytecodes* interpretados por uma máquina virtual [81].

Dentre os principais recursos da linguagem Java destacam-se: orientação a objetos, portabilidade, sintaxe similar a C, extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, segurança, internacionalização, suporte à sistemas distribuídos, desalocação de memória automática, carga dinâmica de código.

O Java 7 é a versão mais recente do Java, detendo diversos novos recursos para aumentar a eficiência de desenvolvimento e execução de programas, sendo portanto utilizado para a codificação da aplicação Fluzz. Fatores como desempenho, estabilidade, segurança, e melhorias na Máquina Virtual Java são algumas das inovações providas por esta última versão da plataforma, que pode ser encontrada em [81].

10.1.2 Eclipse Indigo

O Eclipse é um dos IDEs *(Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) mais populares atualmente para desenvolvimento em plataforma Java, podendo ser adquirido livremente em [98].

Adepta ao *software* livre, a IDE possui facilidades como visualização diferenciada dos arquivos contidos no projeto, ferramentas de gerenciamento de trabalho coletivo, customização do ambiente de trabalho, modos de depuração, editores de texto e gráficos, além de um compilador e interpretador embutidos.

O Eclipse possui um modelo de desenvolvimento baseado em *plugins*, que adicionam funções à ferramenta facilitando a codificação. Trata-se de um ambiente originalmente escrito para ser versátil e possível de ser adaptado para qualquer propósito relacionado ao desenvolvimento de *software*. Sua flexibilidade torna-o configurável perante as necessidades do desenvolvedor, que pode fazer o uso de diferentes perspectivas [98].

Existem várias versões liberadas do Eclipse, cada uma provendo novos recursos. Dentre as melhorias do Eclipse Indigo, utilizado para a construção do Fluzz, a que mais se destaca é o suporte criado para a versão mais recente da plataforma Java, o Java 7.

10.1.3 JADE 3.7

JADE (*Java Agent Development Framework*) é um arcabouço para desenvolvimento de aplicativos baseados em agentes em conformidade com as especificações FIPA, visando a interoperabilidade inteligente de Sistemas Multiagentes. Uma análise minuciosa do *framework* pode ser encontrada no Capítulo 6.

Por apresentar-se de maneira coesa e principalmente bem testada, avaliada e documentada pela comunidade, a versão do JADE utilizada pela aplicação Fluzz foi a 3.7. Apesar de já existirem algumas versões posteriores, no momento da construção da aplicação, a versão 3.7 se mostrou mais estável perante os propósitos do projeto.

A versão 3.7 do JADE foi batizada por seus criadores como *OSGAgents*, devido à integração com OSGi [83], uma tecnologia que fornece um sistema de gerenciamento de componentes de uma forma limpa e formal. Graças a esta integração, tornou-se possível executar agentes JADE dentro de um ambiente OSGi, dando aos agentes acesso a todas as funções típicas da tecnologia, como registrar e usar seus serviços [96].

Outra característica notável introduzida com a versão 3.7 é o suporte para a invocação dinâmica de serviços Web provido pelo WSDC (*Web Service Dynamic Client*).

10.1.4 JUNG 4.2

O JUNG (*Java Universal Network/Graph Framework*) é uma biblioteca de *software* que fornece uma linguagem comum e extensível para a modelagem, análise e visualização de dados passíveis de ser representados como grafos ou redes. Uma análise detalhada da biblioteca é apresentada no Capítulo 7.

A versão do JUNG utilizada para a construção da aplicação Fluzz foi a 2.0.1, que representa a última versão liberada e disponibilizada do *framework*. Esta versão além de conter os recursos mais avançados de visualização gráfica, destaca-se pelas adaptações às importantes melhorias providas pela plataforma Java, como o uso de *Generics*, e dos padrões de projeto *Factory* e *Transformer*, além de possibilitar a modelagem de grafos através do uso de classes representativas para vértices e arestas.

Todos estes avanços foram fundamentais para simplificar e flexibilizar a codificação através do JUNG, que indubitavelmente se tornou um robusto *framework*, fortemente utilizado para a análise e representação de dados em rede.

10.1.5 PostgreSQL 9.0.1

O PostgreSQL é um sistema gerenciador de banco de dados objeto relacional desenvolvido como projeto de código aberto. Atualmente, o PostgreSQL é um dos SGBDs (Sistema Gerenciador de Bancos de Dados) mais avançados, contando com recursos como consultas complexas, chaves estrangeiras, integridade transacional, controle de concorrência multiversão, suporte ao modelo híbrido objeto-relacional, visões, gatilhos, e recursos para criação de procedimentos armazenados, podendo ser adquirido em [99].

Criado em 1985 possui uma arquitetura que ganhou forte reputação pela confiabilidade e integridade de dados provida. O PostgreSQL funciona em todos os principais

sistemas operacionais, sendo totalmente compatível com as propriedades ACID, acrônimo de Atomicidade, Consistência, Isolamento e Durabilidade.

O SGBD inclui vários tipos de dados, além de permitir o armazenamento de objetos binários grandes, como imagens, sons ou vídeos. Possui interfaces de programação nativas para C/C++, Java, .Net, ODBC, entre outros, e documentação atualizada [99].

A versão 9.0.1 do PostgreSQL foi a utilizada para a implementação da estrutura de dados da aplicação Fluzz. Esta versão possui avanços em segurança, suporte a aplicações, monitoramento, performance, armazenamento de dados especializados, e novas capacidades de replicação que aceleram a adoção em hospedagem na nuvem.

O PostgreSQL 9.0.1 inclui ainda outras melhorias que aprimoram os aspectos do projeto e da performance de aplicações de bancos de dados, incluindo suporte a Windows 64 bits, gatilhos condicionais e por coluna, e restrições de unicidade postergáveis [31].

10.1.6 Sistema Operacional Windows 7

Microsoft Windows é uma família de sistemas operacionais criada pela Microsoft, empresa fundada por Bill Gates e Paul Allen [73]. Antes da versão NT o Windows era somente uma interface gráfica para o sistema operacional MS-DOS. Por ser um produto comercial, possui preços diferenciados para cada uma de suas versões, e atualmente é o sistema operacional mais utilizado em computadores pessoais no mundo [14].

Até a versão 3.11, o sistema rodava somente em 16 bits, limitação superada pelo Windows 95, que atuava com 32 bits. As versões a partir do XP e Server 2003 estão preparadas para a tecnologia 64 bits [73].

Com o codinome Vienna, o Windows 7 é o sucessor do Windows Vista que inclui uma série de novos recursos e melhorias. Lançado em Outubro de 2009, esta versão do Windows manteve-se como a mais recente durante praticamente todo o ano de 2012, sendo então o sistema operacional utilizado para a criação da aplicação Fluzz. Testes também foram realizados nas versões XP e Vista do Windows, que processaram a aplicação perfeitamente.

O Windows 7 foi uma versão focada no objetivo de torná-lo compatível com aplicações e *hardwares* com os quais o Windows Vista mostrava-se incompatível.

Em 2012, o Windows 7 ultrapassou o Windows XP alcançando 46,7% dos usuários mundiais, tornando-se o sistema operacional mais usado do mundo. Este sistema operacional inclui vários novos recursos como melhorias no reconhecimento de escrita e no design visual, suporte para discos rígidos virtuais, melhorias no desempenho de processadores multicore, e do kernel. Além disso, o Windows 7 provê suporte para sistemas com múltiplas placas gráficas de diferentes fornecedores (multiGPU) [73].

10.1.7 Arquitetura MVC (Modelo-Visão-Controle)

O crescimento contínuo dos sistemas requer que seus projetos sejam feitos de maneira planejada e componentizada, de modo a torná-los mais flexíveis para facilitar a manutenção e o desenvolvimento dos mesmos [33]. Com o uso da arquitetura em camadas, é possível separar os dados de persistência, das regras de negócio e da interface do usuário, facilitando o desenvolvimento e a manutenção de *software*.

A arquitetura MVC (Modelo-Visão-Controle) foi criada nos anos 80 na Xerox Parc, tendo como princípio a separação entre as camadas de persistência e apresentação. Para isso, foi criada uma arquitetura com três camadas representativas de diferentes objetos que se comunicam entre si. A arquitetura MVC é atualmente utilizada em diversos *frameworks* de várias linguagens como Java, PHP e .Net, provendo importantes recursos como o reuso de código, por exemplo [33].

A Figura 10.1 [69] representa a Arquitetura MVC decompondo o *software* em suas partes lógicas: o Modelo consiste no nível de dados, a Visão representa o nível de interface de usuário, e o Controle assume o nível de processamento.

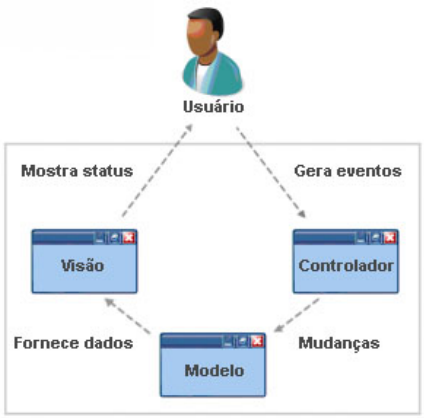


Figura 10.1: *Arquitetura MVC da Aplicação Fluzz.*

A arquitetura da aplicação Fluzz pode ser comparada com um padrão MVC, onde o agentePrincipal representa a camada de visão, a Sociedade de Agentes de Pesquisa atua como a camada controladora, e o Repositório de Dados assume a camada de persistência.

10.1.8 Configuração de *Hardware*

Para a criação, compilação, e processamento de todos os recursos da aplicação Fluzz foi utilizada a seguinte configuração de *hardware* apresentada na Tabela 10.1.

Tabela 10.1: *Configuração de Hardware*

Tipo de Computador: ACPI x64-based PC	Placa Mãe: Asus P5VD2-X
Tipo de processador: DualCore Intel Core 2 Duo E6300, 1866 MHz (7 x 267)	Tamanho da Memória: 2,94 GB
Disco rígido: ST3200820A ATA Device (200 GB, 7200 RPM, Ultra-ATA/100)	Adaptador gráfico: NVIDIA GeForce 7100 GS (128 MB)

10.2 Implementação dos Requisitos

Esta seção tem com objetivo especificar como cada requisito do projeto foi implementado, expondo para isto as técnicas de programação empregadas. Para a completa identificação destes requisitos foi criada a seguinte legenda: RF (Requisito Funcional) e RNF (Requisito Não Funcional), que será então seguida da denominação original dos mesmos apresentada no capítulo anterior.

Antes da solução dos requisitos, é apresentado na Figura 10.2 o *layout* da aplicação Fluzz, produzido automaticamente pelo agentePrincipal na inicialização do sistema. Cinco painéis compõem este *layout*: o superior, o inferior, o esquerdo, o direito, e o central, cada um com algum grupo de funcionalidades definidas.

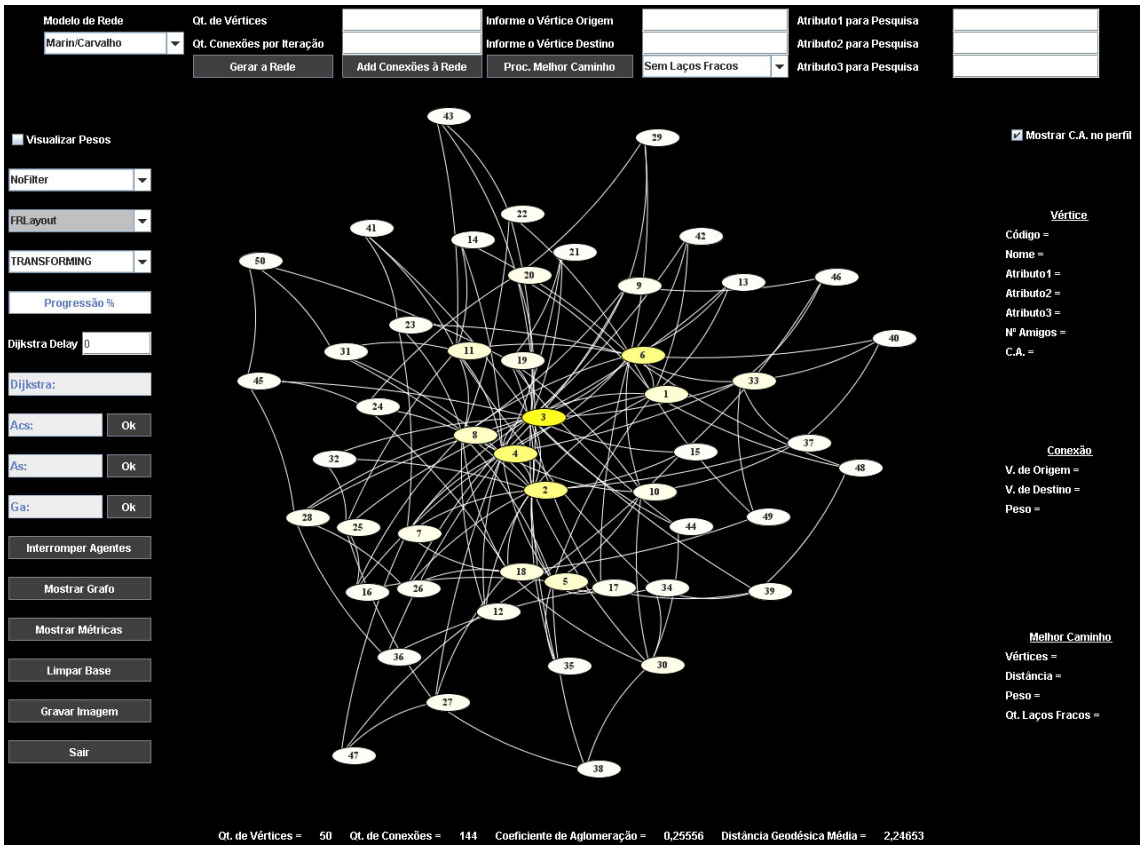


Figura 10.2: *Layout da Aplicação Fluzz.*

A classe que ativa o agentePrincipal é a classe *Principal*, visualizada no código 10.1. Inicialmente, esta classe cria todos os componentes da interface. Em seguida, uma instância da classe *CarregaGrafo* é criada, para que os dados salvos no banco de dados possam ser carregados. Finalmente, o método *iniciaAgentePrincipal()* é invocado, para que a classe *Principal* possa ser registrada como um agente JADE. Será a partir desta interface produzida por este agente, que o usuário poderá interagir com a aplicação, usufruindo de todos os seus recursos e funcionalidades descritos individualmente a seguir.

Código 10.1 Inicialização da Aplicação Fluzz

```
1  public class Principal extends Agent {
2   public static void main(String[] args) {
3    ... // criação de componentes da interface
4    new CarregaGrafo();
5    iniciaAgentePrincipal();
6   }
7   private static void iniciaAgentePrincipal() {
8    String[] param = new String[2];
9    param[0] = "";
10   param[1] = "agentePrincipal:Principal";
11   Boot.main(param);
12  }
13 }
```

10.2.1 RF: Persistência dos dados gerados em um banco de dados

Através da modelagem da estrutura de dados realizada no projeto da aplicação Fluzz, e da determinação do SGBD a ser utilizado, a descrição da implantação desta modelagem pôde ser realizada, conforme exemplifica o Código 10.2.

Código 10.2 Criação do Banco e das Tabelas do Fluzz no PostgreSQL

```
1  Create Database fluzz
2  Create Table nos
3  (
4    id integer not null,
5    nome character varying(100) not null,
6    atributo1 character varying(100) null,
7    atributo2 character varying(100) null,
8    atributo3 character varying(100) null,
9    Primary Key (id)
10 )
11 Create Table conexoes
12 (
13   id_1 integer not null,
14   id_2 integer not null,
15   Primary Key (id_1, id_2),
16   Foreign Key (id_1) References nos (id),
17   Foreign Key (id_2) References nos (id)
18 )
```

Ao término do processo de geração de redes, ou do processo de adição de conexões aos membros das redes, os dados criados (vértices e arestas) serão persistidos no banco de dados *fluzz*, definido na implementação do Código 10.2.

10.2.2 RF: Visualização bidimensional das redes por meio de grafos

O painel central da Figura 10.2 exemplifica o processo de visualização de redes através de grafos bidimensionais realizado pelo Fluzz, obtido pelo clique do botão *Mostrar Grafo* no painel esquerdo da aplicação. Os vértices são representados por elipses, e as arestas por linhas curvas, simbolizando as pessoas e suas conexões respectivamente.

O método que provê a visualização dos grafos é o *mostraGrafo()* pertencente à classe *Principal*. Na linha 4 deste método apresentado no Código 10.3, é criado inicialmente um *layout* específico de acordo com a instanciação de uma das classes de *layouts*, que recebe um grafo como parâmetro. A seguir, um componente de visualização é gerado recebendo o *layout* criado anteriormente, como mostra as linhas 5 e 6 deste mesmo código. Por fim, adiciona-se o componente ao *Frame* para sua visualização.

Código 10.3 Visualizando Grafos no Fluzz

```
1  public class Principal extends Agent {
2   ...
3   private static void mostraGrafo(UndirectedSparseGraph<MyNode, MyLink> grafo) {
4    Layout<MyNode,MyLink> layout = new CircleLayout<MyNode,MyLink>(grafo);
5    VisualizationViewer<MyNode,MyLink> component;
6    component = new VisualizationViewer<MyNode,MyLink>(layout);
7    JFrame frame = new JFrame();
8    frame.add(component);
9    ...
10   }
11  }
```

Alguns métodos disponibilizados pelo JUNG, capazes de realizar mapeamentos visuais, são utilizados por esse componente de visualização de grafos do Fluzz. Dentre eles destacam-se: *setVertexFontTransformer()*, que especifica qual fonte será utilizada para a escrita dos rótulos dos vértices; *setVertexShapeTransformer()*, que especifica a forma com que os vértices devem ser desenhados; *setEdgeFontTransformer()*, que informa a fonte para a escrita do rótulo das arestas; *setEdgeStrokeTransformer()*, que determina a forma como as linhas das arestas serão desenhadas.

10.2.3 RF: Distinção visual dos vértices pela intensidade de cores

Observa-se no grafo da Figura 10.2 que os vértices possuem tonalidades distintas. Vértices menos conectados possuem coloração branca, enquanto que à medida que se tornam mais bem conectados, recebem a coloração amarelo com mais intensidade. O método responsável pela coloração dos vértices é o *setVertexFillPaintTransformer()* do componente de renderização visual do JUNG. No caso do Fluzz, este método possui como parâmetro uma instância da classe *Graph_VertexPaint* apresentada no Código 10.4.

Código 10.4 Coloração dos Vértices do Grafo

```
1  public class Graph_VertexPaint implements Transformer <MyNode, Paint> {
2   ...
3   public Paint transform (MyNode no)      {
4    int valorLim = grafo.getVertexCount()/255;
5    if (valorLim == 0) {
6     valorLim = 1;
7    }
8    int valorCor = 255-((int)Math.pow(grafo.getNeighbors(no).size(),1.7)/valorLim);
9    if (pickedInfo.isPicked(no)) {
10    return Color.CYAN;
11   } else {return new Color(255, 255, valorCor);}
12  }
13 }
```

Implementando a interface *Transformer*, a classe *Graph_VertexPaint* intensifica a quantidade de amarelo aplicada aos vértices à medida que se tornam mais conectados, informando uma instância da classe *Color* como retorno ao método *transform*(), que recebe um vértice como parâmetro. O método utilizado para verificar o grau de conexão dos vértices é o *getNeighbors()*. Já a variável *valorCor* é a responsável por definir a cor exata a ser aplicada. Quando recebe um valor igual a zero atribui coloração máxima de amarelo ao vértice, enquanto que ao atingir o valor 255, torna o vértice branco.

10.2.4 RF: Seleção individual dos vértices e das arestas

No painel esquerdo do *layout* da aplicação Fluzz existe uma opção para a escolha entre dois tipos de seleção de componentes no grafo, sendo ambos os tipos providos pelo *framework* JUNG. O tipo de seleção *PICKING* funciona para a seleção individual de vértices e arestas do grafo gerado, sendo necessário sua escolha quando esta for a necessidade. A Figura 10.3 ilustra a escolha deste tipo de seleção.

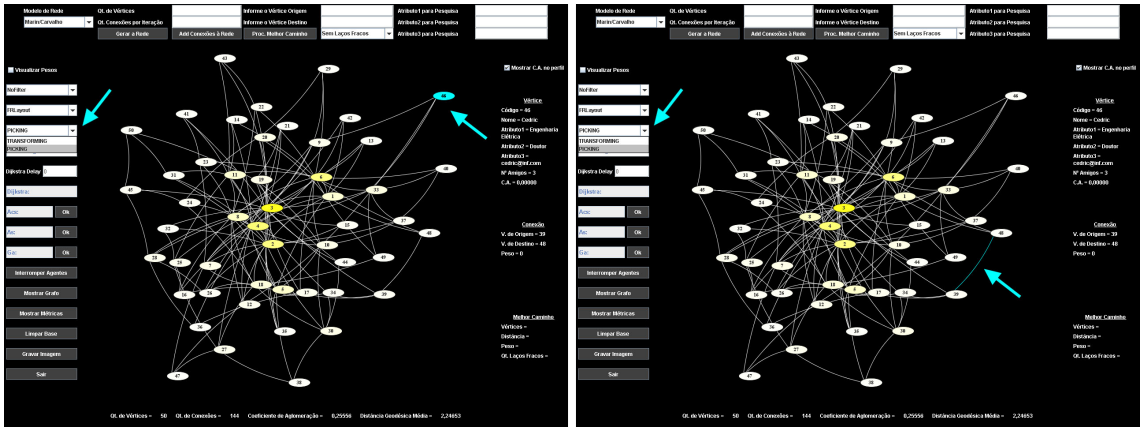


Figura 10.3: *Seleção dos Componentes do Grafo.*

10.2.5 RF: Distinção de cores para vértices e arestas selecionados

No lado esquerdo da Figura 10.3 pode ser verificado que o vértice selecionado assumiu a coloração *Cyan*, enquanto que no seu lado direito, a mesma constatação pode ser realizada para a aresta escolhida. No código 10.4, o método *isPicked()* é o responsável por verificar qual vértice foi selecionado, atribuindo a cor *Cyan* da classe *Color* ao mesmo. Com relação às arestas, a mesma prática é realizada, tendo a classe *Graph_EdgePaint* a mesma função da classe *Graph_VertexPaint*. Obviamente, a única distinção entre ambas, é que as arestas possuirão sempre a cor branca até que uma delas seja selecionada.

10.2.6 RF: Movimentação individual ou em grupo dos vértices

Com o tipo de seleção *PICKING* escolhido na interface da aplicação, pode-se movimentar um único vértice selecionado com o mouse. O lado esquerdo da Figura 10.4 ilustra esta situação, onde um vértice foi movimentado deixando sua posição original.

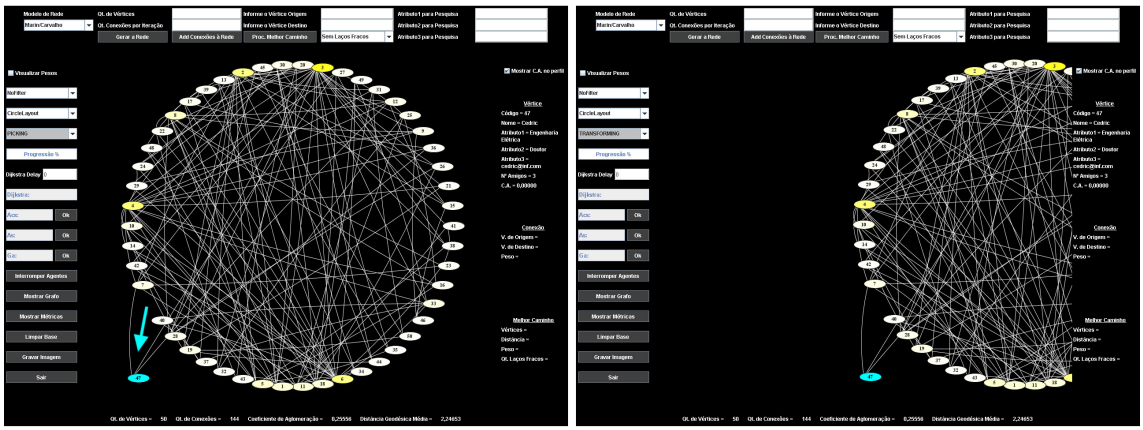


Figura 10.4: *Movimentação dos Componentes do Grafo.*

Caso haja a necessidade de movimentação de vários vértices simultaneamente, é possível criar com o mouse uma área de seleção que abranja mais de um vértice, possibilitando que todos possam ser movimentados em conjunto.

10.2.7 RF: Movimentação do grafo inteiro

Para a movimentação do grafo inteiro, o tipo de seleção *TRANSFORMING* deve ser informado na interface da aplicação. À direita da Figura 10.4 esta situação é ilustrada, onde todo o grafo foi movimentado no painel central, através do uso do mouse.

10.2.8 RF: Variação da escala de visualização do grafo

A Figura 10.5 ilustra a situação em que um mesmo grafo tem sua escala de visibilidade diminuída, como no lado esquerdo da figura, ou aumentada, como no lado

direito da mesma. Estes mecanismos de *zoom* são providos pelo *framework* JUNG, e podem ser utilizados através do botão de *scroll* do mouse. A variação entre a menor e a maior escala de visualização é um dos grandes recursos do arcabouço, que consegue reduzir ou amplificar uma imagem um número considerável de vezes.

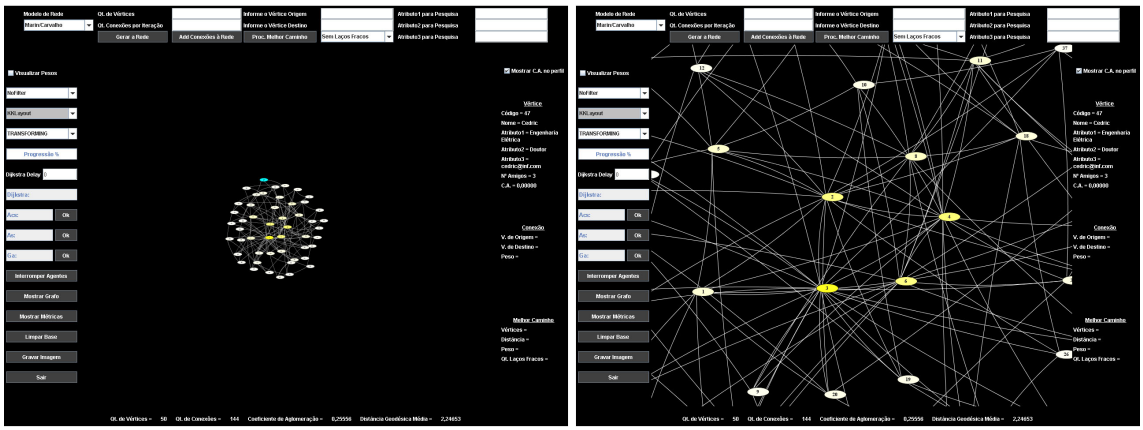


Figura 10.5: *Mecanismos de Zoom do Grafo.*

10.2.9 RF: Rotação do grafo em torno de seus eixos

Outro mecanismo provido pelo Fluzz através do *framework* JUNG é a possibilidade de realizar rotação de grafos. Utilizando como base a Figura 10.4, pode-se verificar no lado esquerdo da Figura 10.6 que o grafo foi rotacionado. Para o uso deste mecanismo é necessário, com o tipo de seleção *TRANSFORMING*, manter a tecla *Shift* do teclado pressionada, enquanto se conduz a rotação do grafo com o mouse.

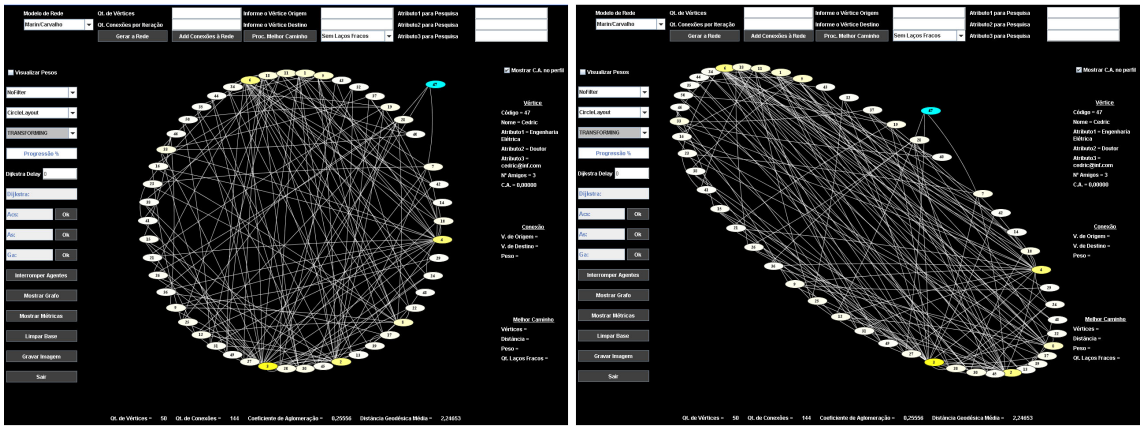


Figura 10.6: *Rotação e Deformação do Grafo.*

10.2.10 RF: Deformação do grafo de maneira uniforme

O Fluzz também provê mecanismos para a deformação de grafos. Mantendo a tecla *Ctrl* do teclado pressionada, e movimentando o mouse, pode-se deformar a rede à esquerda da Figura 10.6, como mostrado à direita da mesma.

10.2.11 RF: Visualização do grafo por meio de diversos *layouts*

O Fluzz provê diversos *layouts* de visualização de redes, conforme pode ser verificado na Figura 10.7.

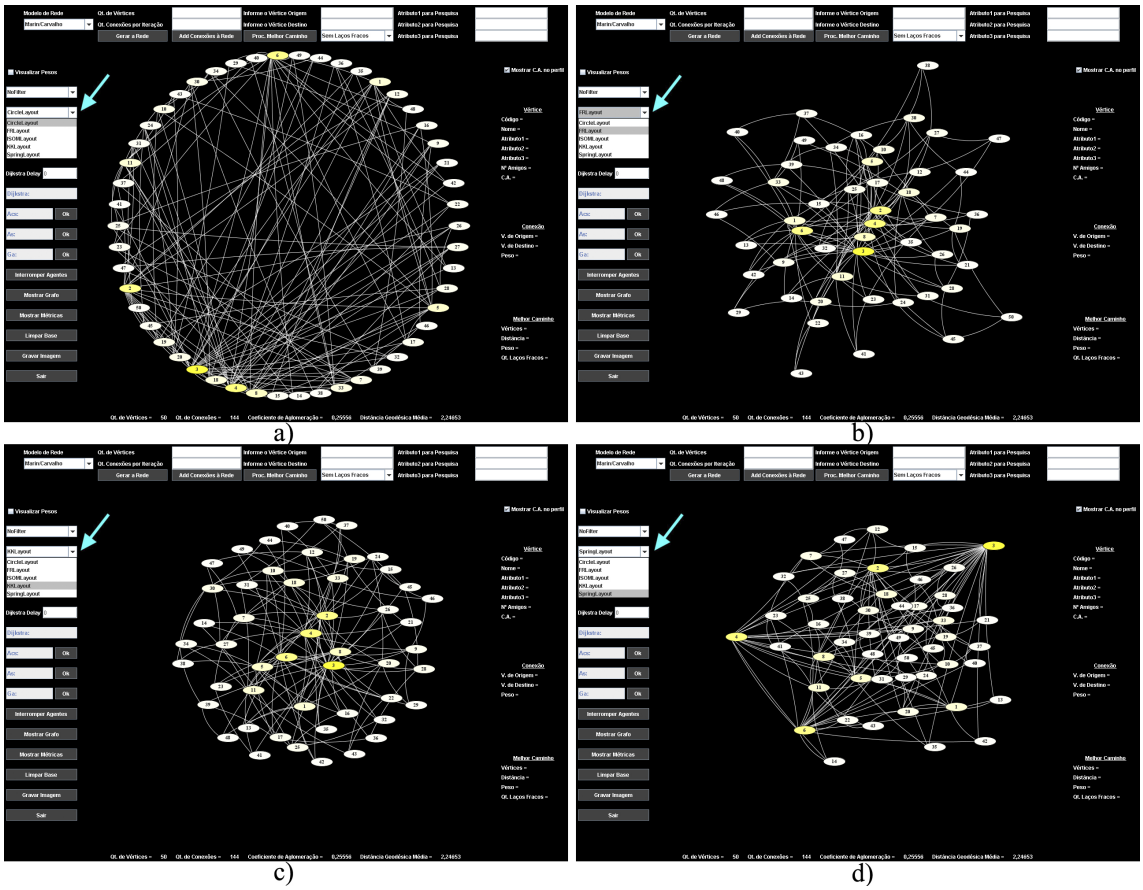


Figura 10.7: *Layouts da Aplicação Fluzz.*

Ao todo são cinco os possíveis *layouts*: *CircleLayout*, *FRLayout*, *ISOMLayout*, *KKLayout*, *SpringLayout*, que podem ser selecionados através de uma opção no painel esquerdo da aplicação, como indicado na figura acima. Cada *layout* utiliza um algoritmo de visualização específico conforme discutido na seção 7.1.3.

Os exemplos ilustrados na Figura 10.7 representam diferentes visualizações da mesma rede. A imagem visualizada na posição (a) desta figura foi criada pelo *layout CircleLayout*. Já a imagem visualizada na posição (b) foi gerada pelo *layout FRLayout*. A imagem visualizada na posição (c) foi criada pelo *layout KKLayout*. Finalmente, a posição (d) da figura apresenta uma imagem gerada pelo *layout SpringLayout*.

O método que realiza a troca de *layouts* é o *mostraGrafo()* pertencente à classe *Principal*. Basicamente, este método apresentado no Código 10.5, verifica qual foi o *layout* selecionado, dentre os cinco possíveis, e o aplica ao componente de visualização de redes da aplicação.

Código 10.5 Alterando *Layouts* de Visualização das Redes

```
1  public class Principal extends Agent {
2   ...
3   private static void mostraGrafo(UndirectedSparseGraph<MyNode, MyLink> grafo) {
4    if (cboTipoLayout.getSelectedItem().equals("CircleLayout")) {
5     Layout<MyNode,MyLink> layout = new CircleLayout<MyNode,MyLink>(grafo);
6    }else if (cboTipoLayout.getSelectedItem().equals("KKLayout")) {
7     Layout<MyNode,MyLink> layout = new KKLayout<MyNode, MyLink>(grafo);
8
9    }
10   ... // testa próximos layouts
11  }
12 }
```

10.2.12 RF: Cálculo automático dos pesos das arestas

Conforme foi estabelecido no projeto da aplicação Fluzz, o peso das arestas deve ser calculado automaticamente pelo sistema. Este cálculo deve considerar a quantidade de amigos em comum entre os vértices diretamente conectados, para poder ponderar cada respectiva aresta. O componente que realiza esta operação é a classe *AmigosComuns*, apresentada no Código 10.6, invocada na inicialização do sistema.

Código 10.6 Cálculo do Peso das Arestas

```
1  public class AmigosComuns {
2   public AmigosComuns(UndirectedSparseGraph<MyNode, MyLink> grafo) {
3    ...
4    Iterator<MyLink> iterEdges = grafo.getEdges().iterator();
5    while (iterEdges.hasNext()) {
6     nContador = 0;
7     edge = iterEdges.next();
8     noOrigem = grafo.getEndpoints(edge).getFirst();
9     noDestino = grafo.getEndpoints(edge).getSecond();
10    Iterator<MyNode> iterAmigosOrigem = grafo.getNeighbors(noOrigem).iterator();
11    Collection<MyNode> listaAmigosDestino = grafo.getNeighbors(noDestino);
12    while (iterAmigosOrigem.hasNext()) {
13     noAmigo = iterAmigosOrigem.next();
14     if (listaAmigosDestino.contains(noAmigo)) {
15      nContador ++;
16     }
17    }
18    edge.setWeight(nContador);
19   }
20  }
```

Na linha 4 deste código, o sistema obtém todas as arestas do grafo através do iterador *getEdges().iterator()*. Em seguida, entre as linhas 7 e 9, a aplicação detecta os vértices de origem e destino para cada aresta identificada. Da linha 10 à 17, é realizado o procedimento de identificação e totalização dos amigos em comum destes vértices, para que por fim, na linha 18, seja atribuído à aresta corrente o valor totalizado.

10.2.13 RF: Visualização dos pesos das arestas de forma opcional

Uma opção no painel esquerdo da aplicação foi criada para a visualização dos pesos da rede, como indicado na Figura 10.8.

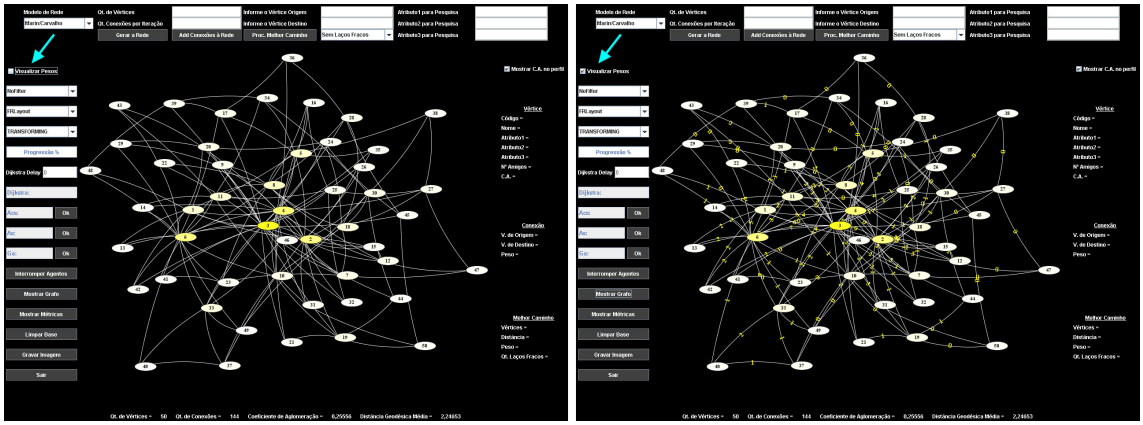


Figura 10.8: *Visualização do Peso das Conexões.*

O método *setEdgeLabelTransformer()* é o responsável por definir como os rótulos das arestas serão impressos. Como parâmetro para o método pode ser repassada uma instância da classe *Graph_EdgeLabelHide*, indicada no Código 10.7. Esta classe recebe uma aresta como argumento e retorna uma *string* nula, o que na prática representa o ocultamento dos rótulos, como visualizado no lado esquerdo da Figura 10.8.

Código 10.7 Escondendo os Rótulos das Arestas

```
1  public class Graph_EdgeLabelHide implements Transformer <MyLink, String> {
2   public String transform(MyLink e) {
3    return null;
4   }
5  }
```

Caso o parâmetro repassado ao método *setEdgeLabelTransformer()* seja uma instância da classe *Graph_EdgeLabelPaint*, indicada no Código 10.8, o retorno da mesma será um documento *HTML* que imprime em amarelo o valor dos pesos da arestas, como mostrado no lado direito da Figura 10.8.

Código 10.8 Apresentando os Rótulos das Arestas

```
1  public class Graph_EdgeLabelPaint implements Transformer<MyLink, String> {
2   public String transform(MyLink l) {
3    return "<html><font color=\"yellow\">" + l.getWeight() + "</html>";
4   }
5  }
```

10.2.14 RF: Visualização do perfil de uma pessoa

No Fluzz, ao se clicar em um vértice específico, os dados referentes ao mesmo são carregados em seu perfil. O perfil fica localizado no painel direito da aplicação conforme ilustrado à esquerda da Figura 10.9. Além das informações referentes às pessoas estipuladas na fase de projeto, foram adicionadas duas métricas calculadas em tempo de execução, o número de amigos e o coeficiente de aglomeração individual, sendo este último apresentado somente se a opção *Mostrar C.A. no perfil* estiver acionada.

Figura 10.9: *Visualização do Perfil de Vértices e Conexões.*

O número de amigos de uma pessoa é calculado pelo método *getNeighborCount()*, que recebe um vértice como parâmetro. Já o coeficiente de aglomeração de um vértice é calculado pela classe *CoeficienteClusterizacao*, como indicado no Código 10.9.

Código 10.9 Cálculo do Coeficiente de Aglomeração

```
1  public class CoeficienteClusterizacao {
2   public double retornaCoeficienteClusterizacao(...Graph... grafo, MyNode no) {
3    return Metrics.clusteringCoefficients(grafo).get(no);
4   }
5   public double retornaCoeficienteClusterizacao(...Graph... grafo) {
6    return Metrics.clusteringCoefficients(grafo);
7   }
8  }
```

A classe *Metrics* do JUNG possui um recurso pronto para o cálculo desse coeficiente, através do método *clusteringCoefficients()*. Este método é sobrecarregado com duas assinaturas, sendo que a primeira delas recebe um grafo e um vértice como parâmetro, para que o sistema possa calcular o coeficiente de aglomeração individual.

10.2.15 RF: Visualização do perfil de uma relação de amizade

Os dados referentes às conexões entre os vértices são também apresentados pela aplicação através de seu painel direito. Clicando-se em uma aresta do grafo, o sistema automaticamente informa quem são os dois vértices que a mantêm, além de apresentar o seu peso referente, como mostrado à direita da Figura 10.9.

10.2.16 RF: Apresentação da quantidade de vértices do grafo

O cálculo do número total de vértices existentes no grafo é obtido pelo método *getVertexCount()* da *API* do JUNG. Esta informação pode ser verificada no painel inferior da interface da aplicação, como pode ser melhor verificado na Figura 10.2.

10.2.17 RF: Apresentação da quantidade de arestas do grafo

Similarmente ao cálculo do número total de vértices de um grafo, o número total de arestas é obtido pelo método *getEdgeCount()*, também da *API* do JUNG. Esta informação fica disponível ao lado da apresentação do número total de vértices.

10.2.18 RF: Apresentação do coeficiente de aglomeração da rede

O coeficiente de aglomeração da rede também é calculado pela aplicação através da classe *CoeficienteClusterizacao*, apresentada no Código 10.9. Para esta operação a segunda assinatura do método *clusteringCoefficients()* é utilizada. Nesta opção, somente o grafo é recebido como parâmetro, e uma média é processada para todos os coeficientes de aglomeração individuais. A informação é disponibilizada no painel inferior da aplicação, e pode ser obtida pelo clique do botão *Mostrar Métricas* situado em seu painel esquerdo.

10.2.19 RF: Apresentação da distância geodésica média da rede

O Fluzz detém recursos para calcular a distância geodésica média da rede, através da classe *DistanciaMedia*, apresentada no Código 10.10. Visualizada no painel inferior da aplicação, esta informação também é obtida pelo clique do botão *Mostrar Métricas*.

Na linha 4 desse código, é criada uma instância da classe *ClosenessCentrality* do JUNG, responsável por prover o método *getVertexScore()*, que informa o inverso

da distância totalizada de um vértice específico para todos os outros vértices do grafo. Neste caso, é necessário inverter este valor para se chegar à distância média em graus de separação, como apresentado na linha 6 do mesmo código. Obviamente, os valores de todos os vértices são calculados, e um valor médio é devolvido pelo método.

Código 10.10 Cálculo da Distância Geodésica Média

```
1  public class DistanciaMedia {
2   public double retornaDistanciaMedia(UndirectedSparseGraph<MyNode, MyLink> grafo) {
3    double dg = 0;
4    ClosenessCentrality<MyNode,MyLink> b = new ClosenessCentrality<MyNode,MyLink>(grafo);
5    for (int i = 0; i < listaNosOrdenados.size(); i ++) {
6     dg += 1/b.getVertexScore(listaNosOrdenados.get(i));
7     contador ++;
8    }
9    return (contador>0)?dg/contador:0;
10   }
11  }
```

Para grafos com uma grande quantidade de vértices, esse cálculo pode ser relativamente demorado, e dependendo da capacidade do *hardware*, o cálculo pode provocar estouro de alocação de memória. Dessa forma, foi criado um procedimento para que este valor possa ser estimado, independentemente do tamanho do grafo, através da coleção *listaNosOrdenados* apresentada na linha 5 do Código 10.10.

Essa coleção é alimentada pelo método *obtemListaNosOrdenada()* da classe *GraphDao*. Inicialmente ele obtém todos os vértices do grafo, criando uma lista de nós ordenada decrescentemente pelos seus graus de conexão, já capturando o primeiro nó desta lista. Em seguida o método divide a lista em 100 partes iguais para poder saltar uma distância relativa ao tamanho da lista divido por 100, e determinar o próximo vértice a ser capturado. Assim, o centésimo vértice escolhido pelo método estará próximo ao último vértice da lista, criando uma boa população para estimar a distância média da rede.

Esse procedimento se mostrou muito mais eficiente do que a escolha aleatória de vértices para a determinação da distância típica dos caminhos, apresentando pequeno desvio padrão. O motivo consiste no fato, de que determinados tipos de topologias pouco igualitárias usualmente teriam suas centenas ou milhares de vértices menos conectados escolhidos mais frequentemente, para integrarem os 100 a serem analisados.

10.2.20 RF: Geração de redes através dos modelos da literatura

Um importante recurso da aplicação é prover meios para a geração automática de redes. O painel superior do Fluzz possui a ferramenta que viabiliza este procedimento. No canto esquerdo deste painel, como assinalado na Figura 10.10, existe uma opção para

a escolha entre os modelos de Erdös e Rényi, Watts e Strogatz e Barabási e Albert, que são os principais modelos para geração de redes propostos na literatura.

Uma vez informado o modelo a ser utilizado, é necessário preencher ainda dois campos para efetivar o procedimento. Primeiro, deve ser informada a quantidade de vértices a ser produzida, e segundo, a quantidade de conexões a ser criada a cada iteração, que corresponde ao número de conexões a ser estabelecido para cada novo vértice gerado. Finalmente, o processo deve ser iniciado através do clique do botão *Gerar a Rede*.

O processo de geração de redes inicia o procedimento criando um grafo completo de tamanho $n + 1$, sendo n a quantidade informada de conexões a ser criada por iteração. Dessa forma, se $n = 2$, por exemplo, o procedimento será iniciado com um grafo de 3 vértices totalmente conectados. A intenção é evitar a geração de grafos não conexos, que impediriam a localização de alguns nós nas redes.

O grafo a ser criado reproduzirá as características essenciais do modelo informado na sua geração. A Figura 10.10 compreende redes geradas pelos modelos citados anteriormente, criadas com 50 vértices, e com duas conexões adicionadas por iteração.

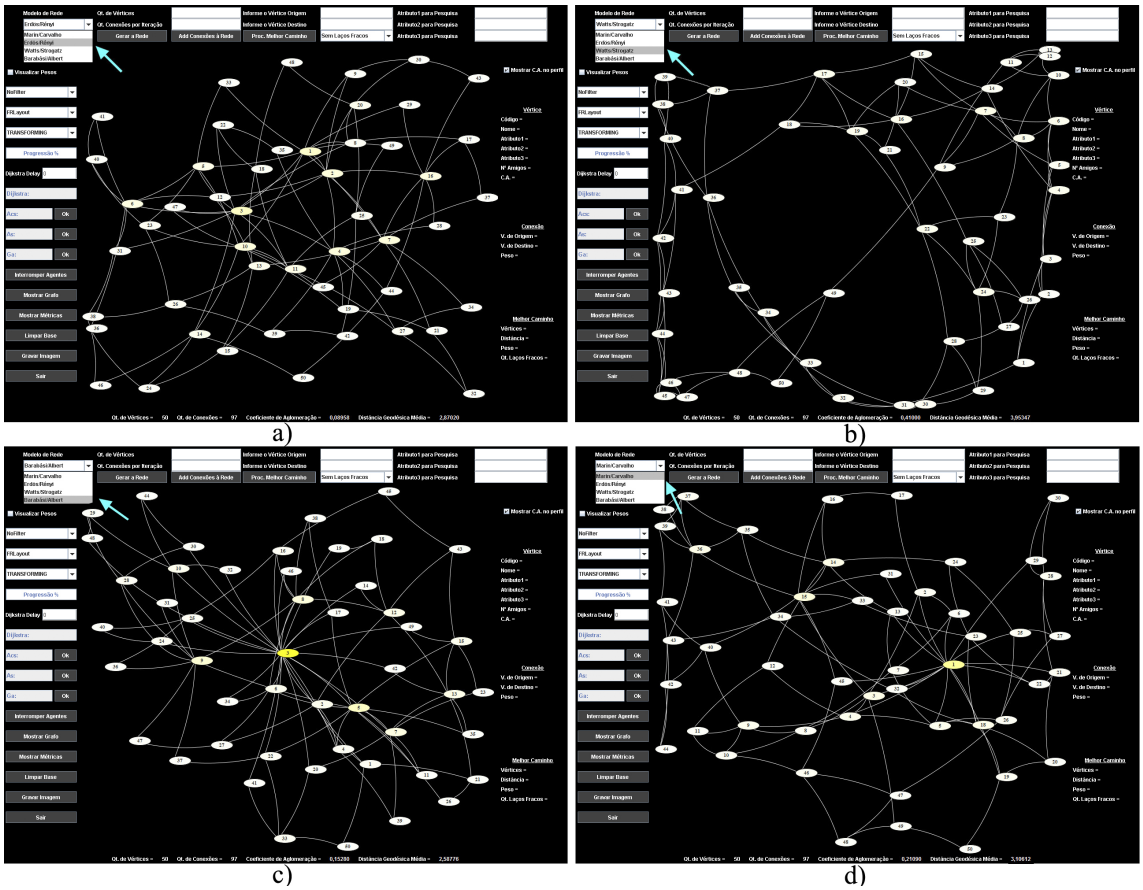


Figura 10.10: *Modelos de Geração de Redes.*

Na posição (a) dessa figura, o grafo foi gerado através do modelo de Erdös e Rényi. O coeficiente de aglomeração apresentado por esta rede foi de 0,089, a distância

geodésica média 2,870, e o vértice mais conectado apresentou 11 conexões. Na posição (b) da mesma figura, a rede foi gerada seguindo o modelo de Watts e Strogatz. Para o coeficiente de aglomeração, o valor encontrado nesta rede foi de 0,410, o comprimento típico dos caminhos foi 3,953, enquanto que o vértice melhor conectado obteve 6 conexões. Já na posição (c) da figura, a rede foi criada pelo modelo proposto por Barabási e Albert. O coeficiente de aglomeração apresentado foi de 0,152, a distância média dos caminhos 2,587, e o vértice mais conectado conseguiu 24 conexões.

Mesmo essas redes sendo relativamente pequenas, já são capazes de apresentar o padrão esperado das quatro características previstas na Tabela 9.1. Redes aleatórias devem possuir aglomeração desprezível, distâncias pequenas entre os vértices, e pouca probabilidade para a emergência de *hubs*. Redes de mundo pequeno já são exímias aglomeradoras, conseguem relativamente encurtar os caminhos, mas são péssimas geradoras de *hubs*. Em contrapartida, redes sem escala são *experts* para a criação destes *hubs*, com alta performance no encurtamento de caminhos, e baixo desempenho na criação de aglomerados.

Apesar de existirem classes disponibilizadas pela *API* do JUNG para a geração de redes de acordo com os três modelos mencionados anteriormente, nenhuma delas pôde ser utilizada pela aplicação Fluzz. O motivo se deve ao fato de que estas classes não possuem recursos para a geração de redes a partir de dados pré-existentes. Portanto, pela *API* do JUNG só é possível criar redes a partir do primeiro vértice.

Consequentemente, não foi possível viabilizar pela API do JUNG o incremento de uma mesma rede, para que a análise das características durante seu crescimento pudesse ser realizada. Além disso, o procedimento para a geração de redes extensas não poderia ser efetivado em um único processamento, pois problemas externos como queda de energia inviabilizariam a operação.

Nesse contexto, foi criada a classe *GeneratorNetwork*, responsável por gerar redes de acordo com todos os modelos mencionados. Esta classe apresentada no Código 10.11 possui um método para criação de conexões de acordo com cada modelo de rede.

Quando instanciada, a classe *GeneratorNetwork* invoca o método *gerarRede()*, repassando os respectivos parâmetros informados. Este método cria uma iteração pelo número de vértices a ser criado, já verificando se a rede possui elementos originais. Neste caso, é atribuído à variável *n* a quantidade de vértices existentes mais 1, como verificado na linha 6 do Código 10.11, e o crescimento da rede é realizado a partir dos dados iniciais.

O procedimento de geração de conexões é o que diferencia todos os modelos. Não obstante, independentemente do modelo, um grafo completo é produzido até que a quantidade de vértices alcance a quantidade de conexões a ser adicionada por iteração mais 1. Em seguida, a adição de conexões seguirá a lógica de cada modelo.

Para o modelo de Erdös e Rényi o método *geraConexoesErdosRenyi()* é invocado, como visualizado na linha 17 do Código 10.11. Ele cria por vértice a quantidade de

conexões informadas no parâmetro seguindo uma única regra: a aleatoriedade.

Já o modelo de Watts e Strogatz é produzido através da invocação do método *geraConexoesWattsStrogatz()*, realizada na linha 19 do Código 10.11. Este método, baseado no padrão de conexões de redes de mundo pequeno, cria em 80% das vezes conexões locais, e para os 20% restantes, conexões aleatórias.

O procedimento de criação de conexões locais segue uma regra particular: encontrar o vértice mais próximo que ainda não esteja conectado ao vértice corrente, utilizando para isto uma avaliação decrescente de seus códigos de identificação. Caso o vértice não seja encontrado até que se chegue ao primeiro elemento da rede, o último elemento será o próximo a ser analisado, como se a rede estivesse disposta em um círculo.

Para atender ao modelo de Barabási e Albert, o método *geraConexoesBarabasi-Albert()* é invocado na linha 21 do Código 10.11. Este modelo também segue uma única regra: a conexão preferencial. Esta regra determina que a probabilidade de um vértice receber uma conexão é proporcional ao seu grau de conexão, o que propicia com a dinâmica evolutiva da rede a possibilidade de surgimento dos *hubs*.

Código 10.11 Geração de Redes

```
1  public class GeneratorNetwork {
2   public GeneratorNetwork(String modelo, int qtNosInformada, int qtConexoesAdicionar) {
3    gerarRede(modelo, qtNosInformada, qtConexoesAdicionar);
4   }
5   public void gerarRede(String modelo, int qtNosInformada, int qtConexoesAdicionar) {
6    n = getGrafo().getVertexCount() + 1;
7    while (n <= qtNosInformada) {
8     adicionaNo();
9     adicionaConexoes(qtConexoesAdicionar, modelo);
10    n ++;
11   }
12  }
13  private void adicionaConexoes(Integer qtConexoesAdicionar, String modelo) {
14   if (getGrafo().getVertexCount() <= qtConexoesAdicionar + 1) {
15         ... // produz grafo completo
16   }else {
17    if (modelo == "Erdös/Rényi") { geraConexoesErdosRenyi(qtConexoesAdicionar);
18    }else if (modelo == "Watts/Strogatz") {
19      geraConexoesWattsStrogatz(qtConexoesAdicionar);
20    }else if (modelo == "Barabási/Albert") {
21      geraConexoesBarabasiAlbert(qtConexoesAdicionar);
22    }
23   }
24  }
25 }
```

10.2.21 RF: Geração de redes de acordo com um novo modelo

A fim de atender ao requisito de criação de um novo modelo para geração de redes, foi implementado na aplicação Fluzz o *Modelo Gama*, definido na fase de projeto da aplicação. O Modelo Gama interpola redes de mundo pequeno e redes sem escala, na esperança de uniformizar positivamente quatro características fundamentais de redes complexas: o coeficiente de aglomeração, o comprimento típico dos caminhos, a possibilidade de geração de *hubs*, e a distribuição de graus.

Uniformizar positivamente significa corrigir pontos fracos, e estabelecer valores aceitáveis para cada característica, minimizando as fraquezas dos modelos. Perante este propósito, redes de diversos tamanhos e densidades foram geradas segundo os três modelos da literatura, e de acordo com três configurações do Modelo Gama. A Tabela 10.2 apresenta os valores obtidos por estes modelos, diante do crescimento da rede original.

Tabela 10.2: *Valores Obtidos Pelos Modelos de Rede*

	E/R			W/S			B/A			MG-40			MG-50			MG-60		
Qt. (V/A)	CA	DGM	MH	CA	DGM	MH	CA	DGM	MH	CA	DGM	MH	CA	DGM	MH	CA	DGM	MH
100/2	0.03	3.37	13	0.37	4.57	6	0.12	3.02	19	0.23	3.81	11	0.13	3.53	13	0.15	3.51	12
100/5	0.12	2.28	23	0.47	2.56	15	0.24	2.20	41	0.34	2.41	17	0.28	2.34	24	0.19	2.29	25
100/10	0.24	1.86	39	0.57	2.02	25	0.35	1.84	66	0.44	1.92	30	0.36	1.89	36	0.32	1.87	38
200/2	0.02	3.79	14	0.36	5.30	7	0.05	3.33	27	0.19	4.23	15	0.10	4.08	16	0.11	3.95	14
200/5	0.07	2.59	27	0.45	3.01	15	0.15	2.44	56	0.31	2.76	24	0.22	2.69	26	0.15	2.63	30
200/10	0.13	2.09	50	0.52	2.37	27	0.22	2.03	97	0.35	2.20	33	0.28	2.14	44	0.24	2.13	46
500/2	0.01	4.48	19	0.35	6.41	8	0.03	3.79	41	0.19	5.00	22	0.11	4.67	21	0.07	4.61	18
500/5	0.02	2.93	32	0.45	3.63	17	0.07	2.74	92	0.29	3.21	30	0.21	3.12	32	0.14	3.03	39
500/10	0.06	2.43	61	0.50	2.75	30	0.12	2.33	155	0.31	2.56	42	0.24	2.51	56	0.17	2.48	62
1000/2	0.00	4.91	20	0.33	7.14	10	0.03	4.15	57	0.19	5.60	25	0.12	5.18	26	0.07	5.08	21
1000/5	0.01	3.25	33	0.44	4.00	17	0.01	3.63	400	0.25	4.59	57	0.18	4.38	67	0.12	3.30	49
1000/10	0.03	2.65	67	0.48	3.02	31	0.07	2.54	212	0.30	2.80	52	0.23	2.74	69	0.15	2.69	83
2000/2	0.00	5.37	22	0.34	8.02	10	0.02	4.33	87	0.19	6.03	26	0.12	5.70	30	0.07	5.46	27
2000/5	0.01	3.54	39	0.44	4.37	19	0.02	3.19	184	0.26	3.86	43	0.18	3.68	46	0.12	3.60	66
2000/10	0.01	2.85	72	0.48	3.34	32	0.05	2.71	282	0.29	3.05	62	0.21	2.97	79	0.15	2.90	101
5000/2	0.00	5.89	23	0.34	9.09	11	0.01	4.72	140	0.18	6.81	33	0.13	6.32	37	0.07	6.05	36
5000/5	0.00	3.90	42	0.44	4.92	20	0.01	3.44	277	0.26	4.30	49	0.19	4.10	56	0.18	3.94	81
5000/10	0.01	3.15	89	0.47	3.71	34	0.02	2.91	463	0.28	3.39	78	0.20	3.28	101	0.14	3.21	134
10000/2	0.00	6.35	24	0.33	9.69	12	0.00	5.00	198	0.18	7.34	41	0.12	6.81	46	0.07	6.44	43
10000/5	0.00	4.17	47	0.43	5.37	22	0.01	3.63	400	0.25	4.59	57	0.18	4.38	67	0.11	4.25	103
10000/10	0.00	3.39	94	0.47	3.40	38	0.01	3.04	649	0.28	3.63	90	0.20	3.52	126	0.14	3.46	169
20000/2	0.00	6.86	27	0.33	10.42	12	0.00	5.25	295	0.19	7.81	48	0.12	7.41	55	0.07	6.78	54
20000/5	0.00	4.44	54	0.44	5.70	23	0.00	3.84	558	0.25	4.90	69	0.18	4.65	81	0.12	4.48	130
20000/10	0.00	3.59	98	0.46	4.28	39	0.01	3.17	901	0.28	3.86	106	0.20	3.75	148	0.13	3.65	208
50000/2	0.00	7.38	27	0.33	11.40	12	0.00	5.61	472	0.18	8.52	61	0.12	7.85	79	0.07	7.34	72
50000/5	0.00	4.79	64	0.44	6.24	23	0.00	4.07	870	0.26	5.28	83	0.18	5.05	102	0.11	4.84	176
50000/10	0.00	3.85	112	0.46	4.67	40	0.00	3.44	1451	0.28	4.18	132	0.20	4.05	185	0.13	3.91	282
100000/2	0.00	7.76	29	0.33	11.38	13	0.00	5.83	692	0.18	8.98	72	0.12	8.31	94	0.07	7.72	87
100000/5	0.00	5.06	69	0.43	6.64	26	0.00	4.31	1231	0.26	5.61	92	0.18	5.32	120	0.12	5.1	224
100000/10	0.00	4.08	118	0.46	4.97	42	0.00	3.59	2031	0.28	4.42	158	0.20	4.25	216	0.13	4.13	345
200000/2	0.00	8.15	31	0.33	12.58	14	0.00	6.11	993	0.18	9.54	77	0.12	8.78	108	0.07	8.14	102
200000/5	0.00	5.30	73	0.44	6.99	27	0.00	4.48	1762	0.26	5.88	111	0.18	5.62	154	0.12	5.37	295
200000/10	0.00	4.29	129	0.46	5.21	44	0.00	3.77	2869	0.28	4.66	179	0.20	4.50	252	0.13	4.35	414
500000/2	0.00	8.69	32	0.33	13.58	15	0.00	6.47	1560	0.18	10.01	88	0.12	9.24	134	0.07	8.52	131
500000/5	0.00	5.65	76	0.44	7.41	30	0.00	4.68	2722	0.26	6.30	130	0.18	5.95	185	0.11	5.67	386
500000/10	0.00	4.57	141	0.46	5.55	47	0.00	3.90	4453	0.28	4.92	213	0.20	4.75	331	0.13	4.63	549

E/R = Modelo de Erdös e Rényi, W/S = Modelo de Watts e Strogatz, B/A = Modelo de Barabási e Albert, MG-40 = Modelo Gama com 40% de conexão preferencial, MG-50 = Modelo Gama com 50% de conexão preferencial, MG-60 = Modelo Gama com 60% de conexão preferencial, Qt. (V/A)= Quantidade total de vértices e quantidade de arestas criadas para cada novo vértice adicionado, CA=Coeficiente de Aglomeração, DGM=Distância Geodésica Média, MH=Maior Hub.

Cada configuração do Modelo Gama definiu a fronteira entre as conexões preferenciais e as conexões locais. A configuração MG-40, por exemplo, assumiu que a probabilidade de ocorrerem conexões preferenciais é de 40%, enquanto que o restante será destinado para conexões locais. A configuração MG-50 determina a mesma probabilidade para ambos os tipos de conexão. Por fim, a configuração MG-60, assume que a maioria das conexões, 60%, será realizada de maneira preferencial.

Durante as mensurações, foi verificado que valores abaixo de 40% de conexões preferenciais inviabilizam a criação de *hubs*. Em contrapartida, valores acima de 60% prejudicam consideravelmente a aglomeração da rede. Por esta razão, as configurações do Modelo Gama utilizadas para a determinação do melhor ponto de ajuste entre os dois tipos de conexão, local e preferencial, ficaram entre os extremos de 40% e 60%.

A variação das configurações do Modelo Gama MG-40, MG-50 e MG-60 influencia na posição inicial, no eixo das ordenadas, da curva de distribuição de graus dos modelos, conforme foi previsto pelas imagens intermediárias da Figura 9.1. A explicação é que quanto mais conexão preferencial existir, mais alto neste eixo a curva se inicia. Além disso, esta variação influi também na velocidade de decaimento da curva, que fica mais branda à medida que se torna mais conectada preferencialmente.

A partir da análise minuciosa dos dados da Tabela 10.2, foi possível observar que a configuração do Modelo Gama MG-50 assumiu a faixa dos melhores valores para a desejada uniformização positiva das características observadas de redes complexas, sendo curiosamente, a configuração mais equilibrada entre os dois tipos de conexão.

Obviamente, o processo de geração de redes pode variar ao longo de sua execução, fazendo com que as configurações conectivas se alternem de acordo com diversas variáveis que envolvem os indivíduos e o ambiente. Não obstante, a configuração MG-50 refletiu na média o melhor intervalo para estas variações, corrigindo os pontos fracos de cada modelo de rede, e ainda mantendo valores relevantes para cada característica.

Ao se encaixar o histograma da conectividade de um nó do Modelo Gama MG-50 em um gráfico, o resultado foi que a distribuição de *links* nas redes geradas por este modelo segue uma hibridação entre distribuições normais e leis de potência, assumindo na maioria das vezes, um padrão próximo ao estipulado na situação *d* da Figura 9.1.

A curva se inicia no valor referente à mínima quantidade possível de conexões de um vértice. Em seguida, sobe seguindo uma distribuição normal até seu ponto máximo, que será aproximadamente o valor inicial mais o valor intermediário da quantidade de conexões a ser adicionada por iteração a cada vértice. Finalmente, a partir deste cume, a curva desce implacavelmente pela restante maioria dos dados da rede, seguindo uma lei de potência com um expoente próximo a 4.

Por esta propriedade de mesclar dois tipos diferentes de redes, apresentando uma pequena região disposta por uma distribuição normal e o restante por leis de potência, o Modelo Gama MG-50 definiu um novo tipo de rede denominado *rede de pequena escala*, que na interface do Fluzz pode ser escolhido através da opção Marin/Carvalho.

A Tabela 10.3 apresenta o comparativo entre a análise das características das redes de pequena escala perante os outros modelos de rede analisados. Percebe-se que este novo tipo de rede não apresenta pontos fracos, e consegue colocar aglomeração, caminhos curtos e *hubs* sob a mesma égide. Portanto, as redes de pequena escala tornam-

se uma alternativa concreta para a definição da real topologia da sociedade.

Tabela 10.3: *Avaliação dos Modelos de Geração de Redes*

	CA	DGM	MH	DG
Redes Aleatórias	0	4	2	Normal
Redes de Mundo Pequeno	5	2	1	Normal
Redes Sem Escala	1	5	5	Leis de Potência
Redes de Pequena Escala	3	3	3	Híbrida (Normal e Leis de Potência)

CA=Coeficiente de Aglomeração, DGM=Distância Geodésica Média, MH=Maior Hub, e DG=Distribuição de Graus.

A posição (d) da Figura 10.10 exemplifica a utilização do modelo de redes de pequena escala. Para 50 nós, com duas conexões adicionadas por iteração, o coeficiente de aglomeração resultante foi de 0,210, a distância média dos caminhos 3,106, e o vértice mais conectado conseguiu 15 conexões, seguindo o padrão estabelecido na Tabela 10.3. Para isso, foi incluído no método *adicionaConexoes()* da classe *GeneratorNetwork*, a instrução referente a esse novo modelo de rede, conforme verificado no Código 10.12.

Código 10.12 `Instrução para Geração de Redes de Pequena Escala`

```
1 public class GeneratorNetwork {
2   ...
3   private void adicionaConexoes(Integer qtConexoesAdicionar, String modelo) {
4     ...
5     else if (modelo == "Marin/Carvalho") {
6       geraConexoesMarinCarvalho(qtConexoesAdicionar);
7     }
8   }
9 }
```

10.2.22 RF: Importação dos dados de redes pré-definidas

Como já foi apresentado, a aplicação Fluzz possui uma estrutura de dados criada fisicamente no banco de dados relacional PostgreSQL. Qualquer rede que esteja armazenada nesta estrutura poderá então ser utilizada pela aplicação. O usuário pode alimentar esta base de dados de três formas: manualmente, utilizando os procedimentos de geração de redes da própria aplicação, ou ainda através de algum aplicativo externo, que possa popular a base adequadamente.

Independente de como a base foi populada, na inicialização da aplicação, uma instância da classe *CarregaGrafo* é criada. Esta última, por sua vez, instancia a classe *GraphDao*, apresentada no Código 10.13, invocando o método *obtemGrafo()*. Este método invoca sequencialmente os métodos *carregaNos()* e *carregaConexoes()*, responsáveis por ler no banco os dados outrora armazenados, que serão posteriormente representados graficamente na interface da aplicação pelo agentePrincipal.

Código 10.13 Montando o Grafo

```
1  public class GraphDao {
2   private static UndirectedSparseGraph<MyNode, MyLink> grafo;
3   private MyNode no;
4   public void obtemGrafo() throws Exception {
5    carregaNos(conexao);
6    carregaConexoes(conexao);
7   }
8   private void carregaNos(Connection conexao) throws SQLException {
9    grafo = new UndirectedSparseGraph<MyNode, MyLink>();
10   stmt = conexao.prepareStatement("Select * from nos");
11   ResultSet rs = stmt.executeQuery();
12   while (rs.next()) {
13    no = new MyNode();
14    ... // atualiza dados do vértice
15    grafo.addVertex(no);
16   }
17  }
18  private void carregaConexoes(Connection conexao) throws SQLException {
19   stmt = conexao.prepareStatement("Select * from conexoes");
20   ResultSet rs = stmt.executeQuery();
21   while (rs.next()) {
22    ...
23    grafo.addEdge(new MyLink(), rs.getInt("id_1"), rs.getInt("id_2"));
24   }
25  }
26  ...
27 }
```

Inicialmente na linha 2 de seu código, a classe *GraphDao* declara um grafo esparso não direcionado, definindo que seus vértices serão objetos da classe *MyNode*, e que suas arestas serão objetos da classe *MyLink*. Estas duas últimas classes possuem todos os atributos modelados para vértices e arestas na fase de projeto.

O método *carregaNos()* será responsável pela leitura dos vértices, que são obtidos através de uma consulta no banco de dados, como mostrado entre as linhas 10 e 16 do Código 10.13. Durante as iterações, uma instância da classe *MyNode* é criada para cada vértice encontrado, que possui seus atributos devidamente atualizados antes de ser adicionado ao grafo.

Quando todos os vértices são adicionados, o método *carregaConexoes()* é invocado para a leitura das conexões existentes. O procedimento é similar ao de criação de vértices, com o detalhe da necessidade de se informar quais são os dois vértices que receberão a conexão. Dessa forma, o procedimento segue até que todo o grafo seja criado.

10.2.23 RF: Adição de conexões às redes existentes

Dentro do contexto de geração de redes, foi criado outro importante recurso na aplicação que provê um mecanismo para o aparecimento de novas ligações entre os vértices existentes. No painel superior do Fluzz existe um botão *Add Conexões à Rede*, responsável por iniciar o processo de criação destas conexões. Para isso, deve ser informada a quantidade de conexões por vértice a ser adicionada, que seguirá um dos modelos de geração de redes disponibilizados pela ferramenta: Erdös/Rényi, Watts/Strogatz, Barabási/Albert ou Marin/Carvalho.

A classe responsável por realizar esse procedimento é a *GeneratorEdges*, e seu funcionamento é bastante semelhante ao da classe *GeneratorNetwork*. A única diferença, basicamente, é que a *GeneratorEdges* não adiciona novos vértices, se concentrando somente na adição de arestas entre os mesmos, como apresentado no Código 10.14.

Código 10.14 Adicionando Conexões às Redes

```
1  public class GeneratorEdges {
2   ...
3   public GeneratorEdges(String modelo, Integer qtConexoesAdicionar) {
4    gerarRede(modelo, qtConexoesAdicionar);
5   }
6   public void gerarRede(String modelo, Integer qtConexoesAdicionar) {
7    Iterator<MyNode> iterVertices = GraphDao.getMapaNosBanco().values().iterator();
8    while (iterVertices.hasNext()) {
9     setNo(iterVertices.next());
10    adicionaConexoes(qtConexoesAdicionar, modelo);
11   }
12  }
13 }
```

A adição de novas arestas entre os vértices das redes invariavelmente produz o aumento da densidade das mesmas. O lado esquerdo da Figura 10.11 ilustra uma rede com 50 nós, gerada com duas conexões por vértice a cada iteração pelo modelo Marin/Carvalho, que implementa redes de pequena escala. Ao se adicionar mais duas conexões por iteração para os mesmos 50 vértices, percebe-se que a rede tem sua distância geodésica média diminuída de 3,105 para 2,113, e seu coeficiente de aglomeração aumentado de 0,239 para 0,264, como visualizado no lado direito da figura.

Na linha 7 do Código 10.14 são buscados todos os vértices do grafo seguindo a ordem de suas criações originais, para que os mesmos possam ser iterados no procedimento de adição de novas conexões. Em seguida, a cada iteração, um vértice é capturado, sendo as conexões consequentemente adicionadas ao mesmo, de acordo com o modelo de rede informado. O método *adicionaConexoes()* da classe *GeneratorEdges*, invocado

na linha 10 deste código, é similar ao seu homônimo da classe *GeneratorNetwork*, com algumas diferenças pontuais para seu melhor ajuste a esta nova situação.

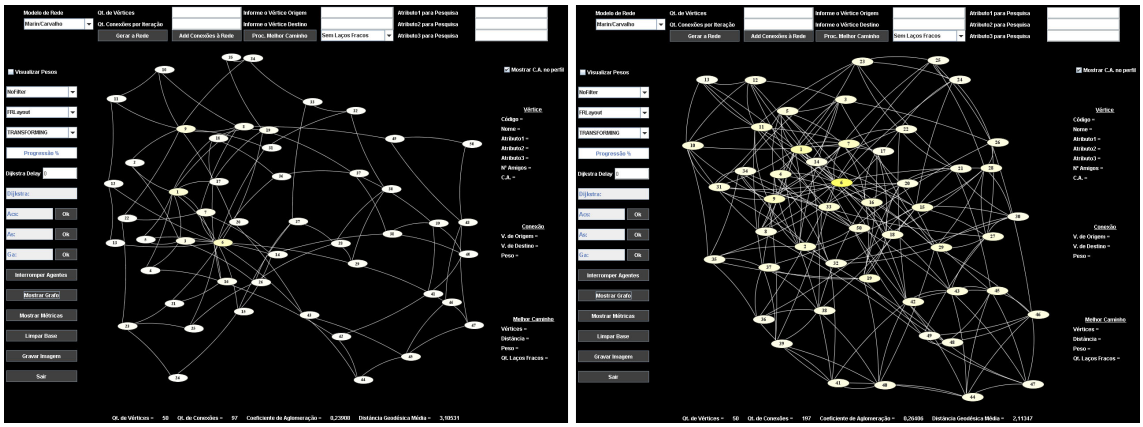


Figura 10.11: *Processo de Adição de Conexões.*

O aumento da densidade provocou essas alterações nos valores mensurados para as principais características de redes complexas. Obviamente, o maior *hub* dessa rede também foi incrementado, passando de 11 para 19 conexões. Todo esse procedimento de adição de conexões entre os vértices possibilita uma diferente análise da rede, que tem seu processo conectivo colocado em foco, para que diversas estimativas e inferências particulares a cada tipo de rede possam ser realizadas.

10.2.24 RF: Detecção de caminhos entre duas pessoas

Além dos recursos de geração e visualização de redes por meio de objetos visuais, a aplicação Fluzz possui adicionalmente um recurso fundamental, que visa atender a outra importante meta do projeto: a realização de buscas.

De acordo com a arquitetura proposta no capítulo anterior, foi construída uma Sociedade de Agentes de Pesquisa, caracterizada por agentes de *software* que trabalham cooperativamente para a resolução de um problema de otimização: minimizar o comprimento dos caminhos, e maximizar o peso das amizades que compõem o caminho existente entre os vértices de origem e destino analisados. A intenção foi de aumentar as chances de interação mútua entre os indivíduos pertencentes aos caminhos nas redes.

Para efetivar o procedimento de busca, o usuário deve informar no painel superior do Fluzz, o código do vértice de origem, de onde a pesquisa se iniciará, e o código do vértice de destino, que é o elemento a ser encontrado na rede. Outra opção, também aceita pela aplicação para a identificação do vértice de destino, é a informação de seu nome descritivo. Em seguida, o botão *Proc. Melhor Caminho* deve ser acionado, e neste momento, o agentePrincipal, representado pela classe *Principal*, inicia os agentes de pesquisa através da invocação do método *criandoAgentes()*, apresentado no Código 10.15.