Contribuições em técnicas analíticas sobre dados representáveis como grafos: delineamento e identificação de padrões para o suporte à decisão

José Fernando Rodrigues Júnior

Texto sistematizando o trabalho científico, apresentado ao Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC/USP), como parte dos requisitos para obtenção do Título de Professor Livre Docente, junto ao Departamento de Ciências de Computação (SCC).

USP – São Carlos/SP
Junho/2016
Este texto foi elaborado para a participação do autor no Concurso Público de Professor Livre Docente junto ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (SCC,ICMC-USP). No texto, é apresentada uma sistematização das pesquisas e dos trabalhos realizados pelo autor na área de Bancos de Dados, com foco em dados representáveis como grafos. Este tema é dividido em três partes: algoritmos para processamento de grafos em larga escala, o que versa sobre técnicas de processamento paralelo assíncrono centrado em vértice para obtenção de desempenho sobre um único nó de processamento; processamento analítico de grafos para detecção de padrões, o que inclui técnicas estatísticas, topológicas, algorítmicas, e algébricas aplicadas sobre propriedades extraídas de grafos, com ampla experimentação sobre a base de dados da Digital Bibliography & Library Project; e, na terceira parte, visualização tendo-se o modelo de grafos como base, na qual usa-se o modelo de grafos tanto para fim como meio para se obter representações gráficas dos dados.

Neste trabalho, aborda-se também o tema de Gerenciamento de Dados baseado em Conteúdo, área em que o candidato também atua de maneira significativa; são descritas metodologias que usam conceitos de Recuperação de Dados baseada em Conteúdo para recuperação, visualização, classificação, e análise da informação. Neste documento, são descritos os fundamentos e os pressupostos com os quais as linhas de pesquisa têm sido exploradas pelo autor, destacando-se os trabalhos e as contribuições nos últimos seis anos de trabalho acadêmico, correspondendo ao período posterior à conclusão do Doutorado.

Palavras-chave: Análise de Dados; Grafos; Recuperação de Dados baseada em Conteúdo; Processamento Paralelo Assíncrono; Visualização; Redes de co-autoria; Redes de Recomendação.
Abstract

This document was elaborated to fulfill the requirements of the author’s application for a position of Associate Professor at the Computer Science Department of the Institute of Mathematics and Computational Sciences, University of São Paulo (SCC, ICMC-USP). The text presents a systematized view of the research and work performed by the author in the field of Databases, with focus on data representable as graphs. This theme is divided in three parts: algorithms for large-scale graph processing, which concerns techniques on vertex-centric asynchronous parallel processing for improved performance over one single processing node; analytical processing of graphs for the detection of patterns, which includes statistical, topologic, algorithmic, and algebraic techniques applied over properties extracted from graphs, with broad experimentation over the database of the Digital Bibliography & Library Project; and, in the third part, visualization based on the graph model, in which graphs are the goal, or the method, to obtain graphical representations of the data. This work also touches on the theme of Content-based Data Management, a field in which the candidate works significantly; the document describes methodologies that use concepts of Content-based Data Retrieval for information retrieval, visualization, classification, and analysis. The text delineates the fundamentals and assumptions with which the research lines have been exploited by the author, with emphasis on the works and contributions of the last six years of academic work, the period after the end of the Doctorial term.

Key words: Data Analysis; Content-based Data Retrieval; Asynchronous Parallel Processing; Visualization; Co-authoring Networks; Recommendation Networks.
Agradecimentos

Todas as pesquisadores e alunos que tiveram parte do desenvolvimento dos trabalhos relatados neste texto foram de fundamental importância para a pesquisa que se sucedeu durante a carreira após o Doutorado do candidato.

Agradeço às agências de fomento Fapesp, Capes, e CNPq pelo apoio financeiro.

Agradeço aos demais professores e funcionários da Universidade de São Paulo pelo apoio nestes anos, em especial à minha eterna orientadora e amiga Agma Traina.

Finalmente agradeço à minha Querida Mãe Ivete Rodrigues, à minha Amada esposa Fabíola Oliveira, e à toda minha família, tão importante em minha vida.
# Sumário

1 Apresentação e introdução
  1.1 Apresentação .............................................. 1
  1.2 Introdução .............................................. 1

2 Compreensão de dados por meio de grafos
  2.1 Considerações iniciais .................................... 5
  2.2 Identificação de fraudes .................................... 6
  2.3 Processamento massivo de grafos .......................... 9
  2.4 Fatoração matricial aplicada sobre a base de dados DBLP 14
  2.5 Técnicas de análise de grafos aplicadas sobre a DBLP 18
  2.6 Sumarização visual de grafos .............................. 23
  2.7 Visualização de grafos em larga escala ..................... 26
  2.8 Visualização de Bancos de Dados Relacionais por meio de grafos 29
  2.9 Hierarchical visual filtering ................................ 32
  2.10 Considerações finais ...................................... 34

3 Gerenciamento de Dados baseado em Conteúdo
  3.1 Considerações iniciais .................................... 37
  3.2 Conceitos sobre Gerenciamento de Dados baseado em Conteúdo 37
  3.3 Visualização de espaços de características .................. 39
  3.4 Otimização do espaço de características ................... 40
  3.5 Detecção de fogo em imagens .............................. 41
  3.6 Detecção de fumaça em imagens ............................ 43
  3.7 Projeto e desenvolvimento de um sistema de Gerenciamento de Dados baseado em Conteúdo 43
  3.8 Considerações finais ...................................... 43

4 Conclusões .............................................. 45

A Identificação de fraudes .................................... 57

B Processamento massivo de grafos .......................... 69

C Fatoração matricial aplicada sobre a base de dados DBLP 81

D Técnicas de análise de grafos aplicadas sobre a DBLP 87

E Técnicas de análise de grafos aplicadas sobre a DBLP 95

F Sumarização visual de grafos .............................. 103
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Visualização de grafos em larga escala</td>
<td>117</td>
</tr>
<tr>
<td>H</td>
<td>Visualização de Bancos de Dados Relacionais por meio de grafos</td>
<td>127</td>
</tr>
<tr>
<td>I</td>
<td>Hierarchical visual filtering</td>
<td>139</td>
</tr>
</tbody>
</table>
2.1 **Esquerda:** Um exemplo de matriz de adjacência (cor azul claro) segundo a técnica *M-Flash* com 3 intervalos lógicos \((β = 3)\). \(G^{(p,q)}\) é um bloco com as arestas no intervalo de origem \(I^{(p)}\) e o intervalo de destino \(I^{(q)}\). **Direita:** Os valores \(γ_i\) dos vértices são armazenados como vetores, os quais são divididos nos mesmos intervalos lógicos dos blocos; por exemplo, na computação do algoritmo *PageRank*, são armazenados o grau de saída dos vértices e os valores intermediários do *PageRank*.

2.2 Exemplo de orden de computação para um grafo com três intervalos. Os blocos são processados em zigzague vertical (sequência de setas vermelha, laranja, e amarela) permitindo o reuso dos dados dos vértices. No exemplo, faz-se reuso dos vértices de destino ao longo da coluna de destino 1, e faz-se reuso dos vértices de origem da linha 3 quando se passa do bloco \(G^{(3,1)}\) para o bloco \(G^{(3,2)}\).

2.3 Exemplo de operações de I/O para processar o bloco denso \(G^{(2,1)}\).

2.4 Etapas do processamento de blocos esparsos.

2.5 Modelo entidade-relacionamento da DBLP.

2.6 Grafo exemplo - relacionamento “professor participa de evento” muitos-para-muitos entre instâncias de professores e instâncias de eventos. Os pesos das arestas indicam o número de participações de um dado professor em um dado evento.

2.7 Decomposição por Valor Singular correspondente ao grafo da Figura 2.6.

2.8 Metodologia de mineração de dados aplicada sobre a DBLP.

2.9 Distribuição autores-artigos. (a) número de artigos \(\times\) número de autores. (b) *Rank-plot* sobre o índice do autor \(\times\) número de artigos.

2.10 Distribuição de *weakly-connected components* (WCC) para co-autoria.

2.11 Average Clustering Coefficient (ACC) \(\times\) grau do vértice para o grafo de co-autoria.

2.12 Distribuição do grau do grafo de co-autoria.

2.13 Evolução ao longo do tempo do *Effective diameter* do grafo de co-edição.

2.14 Sumarização de grafo integrada ao sistema GMine. (a) Grafo original; (b) destaque para um vértice em particular; (c) definição do conjunto de vértices de interesse; (d) subgrafo induzido sumarizando a conectividade relativa aos vértices de interesse.

2.15 O vocabulário de estruturas considerado na metodologia StructMatrix; de (a) a (g), exemplos das estruturas consideradas.

2.16 Esquema da visualização StructMatrix.

2.17 SuperGrafo obtido a partir de um grafo particionado.

2.18 Particionamento dado por percentis estatísticos; neste caso, múltiplos de 5 para o atributo Idade de uma entidade Pessoa.

2.19 SuperGrafo construído a partir do particionamento dado por duas entidades relacionais ligadas por um relacionamento muitos-para-muitos.

2.20 Exemplo de uma visualização hierárquica sobre uma Base de Dados Relacional.

2.21 *Hierarchical Visual Filtering.* (a) Elementos, (b) funções constituintes, e (c) ciclo iterativo.
2.22 Definição de uma análise hierárquica sobre dados da Embrapa. (a) Visualização $V_0$ de todos os dados da relação $D_0$. (b) Filtragem interativa gerando 5 subconjuntos $D_i \subseteq D_0$, $1 \leq i \leq 5$, e correspondentes visualizações $V_i$, $1 \leq i \leq 5$; cada uma com dois conjuntos de parâmetros $s, g$ definindo as visualizações coordenadas paralelas e projeção multidimensional. (c) Novo ciclo iterativo definindo as visualizações $V_i$, $6 \leq i \leq 9$.

3.1 Slim-tree indexando 17 objetos.

3.2 Visualização do resultado de uma consulta por similaridade em um ambiente 3D. (a) Centro da consulta posicionado na origem do espaço Euclidiano de projeção. (b) Interação 3D. (c) Funcionalidade de zoom.

3.3 Conjunto de funções usadas na metodologia proposta (Avalhais et al., 2012).

3.4 Arquitetura FFireDt. O módulo de avaliação recebe uma imagem não rotulada, cria sua representação na forma de um vetor de características, e a classifica usando classificação Instance-Based Learning. O sistema classifica a imagem e provê ao especialista a possibilidade de recuperar imagens semelhantes.

3.5 Exemplo da aplicação da técnica de Superpixel sobre uma imagem contendo fumaça.
Capítulo 1

Apresentação e introdução

1.1 Apresentação

De acordo com alternativa prevista no edital do concurso, descreve-se, neste texto, apenas parte da obra do candidato. Especificamente, o texto descreve somente a atuação em tópicos relacionados à especialidade do concurso. Assim, apenas 15 dos 33 artigos científicos pós-doutorado de sua autoria, ou co-autoria, são reportados. Todos os trabalhos são, no mínimo, 3 anos posteriores à obtenção do título de Doutor. A obra completa do candidato é reportada no texto Memorial Circunstanciado, entregue conjuntamente à documentação exigida.

O texto está estruturado seguindo ao que é pedido no edital, contendo introdução precedendo ao texto sistematizado, e com conclusões a sucedê-lo. Deste modo, a introdução é apresentada na próxima seção (Seção 1.1), o texto sistematizado foi dividido nos Capítulos 2 e 3, e as conclusões são apresentadas no Capítulo 4. A obra do candidato é acompanhada de contextualização e eventual articulação teórica com o intuito de fomentar a compreensão crítica.

1.2 Introdução

A quantidade massiva de dados produzida na era da informação, instituída nos últimos 30 anos (Masuda, 1980)(Kantardzic, 2002), é uma fonte de conhecimento riquíssima; mas é também um problema consolidado que envolve questões de armazenamento, processamento, aproveitamento, e compreensão dos fenômenos subjacentes, questões que são tratadas por diversos ramos da pesquisa em Bancos de Dados. Tais informações são produzidas em muitos aspectos de nossas vidas, desde interações em redes sociais, transações de cartões de crédito, até pegadas digitais produzidas por sistemas de geo-localização. Considerando esta realidade, em 2010, uma publicação do conceituado periódico de notícias The economist...
intitulada *Data, data everywhere* (Economist, 2010) ilustrou as vantagens, como a potencial melhoria da qualidade de vida e dos serviços oferecidos às pessoas; e desvantagens, como problemas de privacidade, que derivam dessa nova magnitude de informação, ressaltando a possibilidade de se utilizar tais dados para melhorar diversos aspectos de interesse geral da sociedade.

Em um estudo recente, publicado em 2014 pelas gigantes multinacionais *General Electric* e *Accenture* (GE e Accenture, 2015), os autores discutem o impacto do uso de técnicas de análise de dados no cenário competitivo industrial, enfatizando a urgência para que as empresas se adaptem para adquirir vantagens de negócio por meio do tratamento adequado dos dados. Além disso, o estudo discute a importância da análise de dados na melhoria de serviços de saúde, envolvendo aspectos clínicos, financeiros, e operacionais. No trabalho de Raghupathi (Raghupathi e Raghupathi, 2014), os autores ressaltam que no passado as decisões tomadas pelos médicos eram mais simples e baseadas principalmente na experiência pessoal, conhecimento da área e avaliação dos sintomas do paciente. Agora com uma quantidade de dados muito maior disponível, tal decisão se torna muito mais complexa e envolve diferentes disciplinas, bem como técnicas eficientes e eficazes para interpretar os dados, auxiliando à tomada de decisão.

Muitos dos dados gerados de maneira ubíqua e constante advêm da chamada Web 2.0, na qual os usuários não são apenas consumidores de conteúdo; eles são, também, geradores de conteúdo que será consumido por outros usuários. Essa dinâmica impulsiona a produção de dados em larga escala, denominada *escala planetária* ou *escala web*. Muitas vezes, esses dados representam relacionamentos entre seus elementos, como em redes de relacionamento, sistemas de recomendação, fóruns online, redes de comunicação eletrônica, redes de citação científica, segurança cibernética, entre muitos outros; dados que são adequadamente representados como grafos. Esses grafos possuem diversas propriedades que podem ser exploradas, como o peso das ligações (podendo representar, por exemplo, a intensidade da conexão), textos associados (o texto da avaliação de um filme feita por um usuário, por exemplo) e dinamismo (novas conexões criadas ao longo do tempo). Comum a estes dados é a larga escala e a complexidade das relações observadas.

Para se analisar grafos em larga escala, os quais podem chegar a bilhões de arestas e vértices, é necessário um alto poder de processamento. Grafos como os que definem os conjuntos de dados experimentais Yahoo Web (Kang et al., 2011), Twitter (Krishnamurthy et al., 2008), e Clickstreams (Liu et al., 2009) são desafiadores; eles não cabem em memória e requerem discos inteiros para serem armazenados – processar tais conjuntos com técnicas usuais (definidas por algoritmos em memória e baseados em varredura de grafo) pode levar semanas. Uma das abordagens para se resolver este problema é usar processamento distribuído organizado em *clusters* computacionais (Kang et al., 2011) gerenciados por arcabouços como o Hadoop (Shvachko et al., 2010); no entanto, abordagens baseadas em *clusters* adicionam complexidade (processamento distribuído) e custos extras (equipamentos e instalações) que podem ser proibitivos. Desta maneira, para se ampliar as possibilidades de processamento de grafos em grande magnitude, surgiram alternativas introduzidas em anos bem recentes a este trabalho. Tais propostas exploram as novas arquiteturas computacionais baseadas em múltiplos núcleos (Han et al., 2013) de processamento centrado em vértice (Kyrola et al., 2012). Estas técnicas permitem que um grafo seja processado de maneira iterativa, gradualmente em passos que cabem em memória, bem como permitem que as propriedades locais dos grafos sejam calculadas sem que seja necessária uma varredura topológica – o que pode ser computacionalmente inviável. Esta linha de pesquisa, o desenvolvimento de técnicas para processamento massivo de grafos em um único nó de processamento, é tema do presente trabalho.
Além disso, como tais grafos possuem características de potencial significado, trabalhos de natureza analítica buscam compreender suas propriedades como a evolução temporal (Aggarwal e Subbian, 2014), enumeração de propriedades (Park et al., 2014), análise de avaliações (Günnemann et al., 2014), análise de sentimentos (Neuenschwander et al., 2014), e decomposição em tensores (Sidiropoulos et al., 2014). Outros trabalhos focam na vertente topológica, a que calcula propriedades como raios (Kang et al., 2011), caminhos mais curtos (Maruhashi et al., 2012), triângulos (Tsourakakis et al., 2011), e cliques (Sakurai et al., 2011); detecção de padrões como comunidades (Yang e Leskovec, 2013); convergência de estados transitórios (Gonçalves et al., 2011); anomalias e fraudes (Akoglu e Faloutsos, 2013). Desta maneira, o processamento analítico de dados representáveis como grafos também é parte da temática tratada nesta obra de Livre Docência.

Além de abordagens algorítmicas e de processamento analítico, grafos também são objetos de trabalho na área de visualização de dados, a qual se baseia em gráficos para apresentar as propriedades calculadas do grafo (Kang et al., 2012) ou sua estrutura (Ma e Muelder, 2013). Em redes que variam no tempo, como redes sociais, de co-autoria, ou da Web, há um desafio adicional. Nestas redes, o grafo cresce constantemente ao passo que novos relacionamentos ocorrem (Sallaberry et al., 2013), gerando redes de altíssima magnitude. Em tais redes, além de padrões que caracterizem a rede globalmente, também é interessante avaliar conexões localmente, pois, nestas redes, um único vértice pode afetar agrupamentos (Correa et al., 2012) ou a evolução da dinâmica global (Tamassia, 2013). Tais linhas de trabalho – visualização global e local de grafos em larga escala – justificam a pesquisa por métodos visual-analíticos inovadores e com bom desempenho; esta linha de trabalho é a terceira temática do presente texto.

O conteúdo a seguir apresenta as principais contribuições do candidato referentes ao tema de análise de bases de dados representáveis como grafos. Como motivado e justificado de maneira circunstanciada nesta seção, os tópicos que serão apresentados versam sobre algoritmos para processamento em larga escala, sobre processamento analítico para detecção de padrões, e sobre visualização tendo-se o modelo de grafos como base. Estes três temas de uma mesma área de pesquisa, embora considerados separadamente para fins de organização, misturam-se quando do desenvolvimento de técnicas e sistemas, como se poderá notar. Estes temas são apresentados no Capítulo 2, o qual perfaz a principal obra descrita neste documento. Adicionalmente, para fins de complementação a respeito do trabalho do candidato após o Doutorado, um capítulo adicional descreve sua atuação no tema de Gerenciamento de Dados baseado em Conteúdo¹ – Capítulo 3. Este capítulo é mais sucinto do que o Capítulo 2 e seus trabalhos não são incluídos no apêndice, mas apenas na documentação complementar do Memorial Circunstanciado. A intenção desta opção de escrita era, ao mesmo tempo, não tornar este documento demasiado extenso, nem desencorajar seu conjunto principal de contribuições.

¹O termo “Recuperação de Dados baseada em Conteúdo” é o mais usado; no entanto, aqui se refere não somente à recuperação de dados, mas também a outras tarefas como a análise e a classificação da informação, de modo que optou-se pelo termo “Gerenciamento de Dados baseado em Conteúdo.”
Capítulo 1. Apresentação e introdução
Compreensão de dados por meio de grafos

2.1 Considerações iniciais

Bases de dados representáveis como grafos estão entre os dados mais amplamente produzidos pela atividade humana do século XXI. Grafos, ou redes complexas\(^1\), surgem em Bancos de Dados Relacionais (Chen, 1976), na telefonia digital (móvel ou não), no uso de redes sociais eletrônicas, no comércio eletrônico, nas redes de computadores, na digitalização de mapas viários, na co-autoria de trabalhos científicos, em cadeias proteicas, dentre muitas outras fontes (Leskovec et al., 2006). Muitos destes dados não são exclusivos da atual configuração tecnológica; no entanto, a capacidade de se coletar tais dados em volume e com precisão só se tornou possível nas últimas décadas.

Tais dados, como tantos outros da era digital, criam uma situação conflitante. Apesar de se poder coletá-los, a capacidade de se fazer uso útil deles com semelhante celeridade ainda carece de avanços. Neste contexto de trabalho, este primeiro capítulo descreve as contribuições alcançadas no aproveitamento de bases de dados representáveis como grafos. Esta é a principal, porém não a única, linha de pesquisa do candidato; ela foi escolhida de modo a se alcançar uma dissertação com tema específico. Não obstante, ao longo do texto, outras contribuições são mencionadas em diferentes graus de detalhamento, sobretudo no Capítulo 3.

Este capítulo, portanto, apresenta a obra principal da Livre Docência, organizada em três linhas de pesquisa:

1. desenvolvimento algorítmico: nesta primeira parte, serão apresentados dois trabalhos; o primeiro, na Seção 2.2, versa sobre um algoritmo paralelo para identificação de fraudes em redes de

\(^1\)Uma rede complexa pode ser entendida como um grafo com topologia não trivial observada em sistemas reais, diferente do que ocorre em grafos aleatórios ou em estruturas trenzadas. Ao longo do texto, os termos grafo, rede, e rede complexa serão usados indistintamente para se referir a bases de dados representadas como grafos.
Capítulo 2. Compreensão de dados por meio de grafos

2. processamento analítico de grafos: na segunda parte, serão apresentados trabalhos que se baseiam em desenvolvimento analítico; na Seção 2.4, será apresentado um trabalho sobre fatoração matricial; na Seção 2.5, descrevem-se dois trabalhos sobre análise baseada em métricas de grafos; e, na Seção 2.6, será apresentada uma técnica de sumarização de grafos;

3. visualização: na terceira parte, serão apresentados trabalhos que usam grafos, mas que têm a apresentação gráfica da informação como objetivo; na Seção 2.7, será apresentada uma técnica de processamento e apresentação de grafos com alta cardinalidade; a Seção 2.8 apresenta um trabalho sobre a visualização dos relacionamentos entre entidades de um Banco de Dados Relacional; e, na Seção 2.9, será apresentado um trabalho sobre uma técnica de interação que define um grafo com o propósito de particionar e organizar os dados de uma relação de uma base de dados.

Nas próximas seções, estas contribuições serão descritas resumidamente. Maiores detalhes, resultados, análises comparativas, e possíveis aplicações são apresentados nos Apêndices A, B, C, D, E, F, G, H, e I. Neste capítulo, e nos demais deste documento, são mantidas a simbologia e a terminologia usadas nas respectivas publicações de maneira a simplificar a consulta aos apêndices.

2.2 Identificação de fraudes

O trabalho Online-Recommendation Fraud Excluder (ORFEL)²(Gimenes et al., 2016) foi desenvolvido para tratar o problema de detecção de fraudes em redes de recomendação. Nestas redes, é possível criar usuários falsos para difamar ou promover produtos por meio de recomendações eletrônicas ilegítimas ao longo do tempo, o que dificulta sobremaneira o problema. Para resolver este problema, usou-se de processamento paralelo assíncrono de maneira a se procurar máximos locais no espaço de possibilidades definido por um conjunto de usuários e um conjunto de produtos. A abordagem gera sementes aleatórias dentro do espaço de recomendações, sendo que cada semente procura por máximos locais em seu entorno. Em processamento assíncrono, as sementes progridem paralelamente, o que permite uma alta escalabilidade, necessária ao problema. Os resultados obtidos mostraram a eficácia da técnica, a qual foi capaz de detectar quase a totalidade dos ataques; os resultados também mostraram a eficiência da técnica, pois ela resolveu o problema com desempenho superior ao da abordagem anterior da literatura, baseada em processamento distribuído.

Dado um conjunto de recomendações usuário-produto no tempo, a técnica ORFEL visa encontrar o que se denomina comportamento lockstep (Beutel et al., 2013); isto é, vários usuários recomendando um ou mais produtos dentro de um período de tempo ∆t. Embora seja semanticamente rico no problema de recomendação usuário-produto, o comportamento lockstep não é exclusivo deste domínio, podendo ser generalizado; para tanto, basta que se tenha um problema modelado como um grafo bipartido, como no caso de redes de citação artigo-revista. Encontrar todas as instâncias deste comportamento é uma modalidade de agrupamento em sub espaço, o que é considerado um problema NP-Difícil (Anagnostopoulos et al., 2008). Assim, ao invés de resolver o problema de maneira ótima, a ideia é

²A técnica ORFEL foi submetida ao periódico Information Sciences (Qualis A1), em Fev/2016; sua concepção se baseou no trabalho de Mestrado do aluno Gabriel Gimenes, orientado do autor deste documento.
2.2. Identificação de fraudes

alcançar uma solução aproximada por meio de processamento paralelo assíncrono centrado em vértice.

Processamento centrado em vértice
Dado um grafo $G = (V, E)$, com vértices rotulados de 1 a $|V|$, arestas $e \in E$, $e = (u, v)$ ($u$ é a origem e $v$ é o destino), e tendo-se valores computáveis associados aos vértices e às arestas, o processamento centrado em vértice (Malewicz et al., 2010) corresponde ao laço descrito no Algoritmo 2.1.

No algoritmo, os valores dos vértices e das arestas são processados de acordo com a computação desejada, ilustrada pelas funções hipotéticas $f$ e $g$. Diferente de algoritmos baseados em varredura, o processamento centrado em vértice não varre a estrutura do grafo, o que reduz o consumo de memória, mas faz com que valores precisem ser propagados ao longo do grafo iterativamente. Assim, uma única passagem do algoritmo 2.1 não é suficiente para computação útil, requerendo que o grafo seja processado tantas vezes quanto necessário para que se alcance um dado critério de convergência específico para cada computação. O processamento de grafo, então, define-se como descrito no Algoritmo 2.2.

Algoritmo 2.1 Processamento de grafo centrado em vértice.

```plaintext
procedure Graph_scan(Graph G)
    for $i = 1$ to $|V|$ do
        $set_e \leftarrow$ conjunto de arestas adjacentes a $V[i]$
        $V[i].value \leftarrow f(set_e)$
        for cada aresta $e \in set_e$ do
            $e.value \leftarrow g(V[i].value, e.value)$
        end for
    end for
end procedure
```

Algoritmo 2.2 Processamento de grafo.

```plaintext
procedure Graph_processing
    while (critério de convergência não é satisfeito) do
        Graph_scan(G)
    end while
end procedure
```

O processamento centrado em vértice contrasta com a usual varredura de grafo, a qual percorre a estrutura do grafo por largura ou por profundidade. Ao passo que algoritmos baseados em varredura suportam qualquer tipo de processamento, tais algoritmos demandam que o grafo esteja todo em memória; do contrário, o necessário acesso aleatório seria inviável ao ser executado em disco. O processamento centrado em vértice suporta o processamento de grafos mesmo que estes não caibam em memória; no entanto, nem todo tipo de processamento pode ser realizado. Esta estratégia suporta apenas problemas que possam ser resolvidos considerando-se a vizinhança dos vértices; ou problemas baseados em varredura, mas que possam ser adaptados para a resolução por vizinhança – exemplos incluem PageRank (Page et al., 1999), análise espectral (Kang et al., 2014), estimativa de diâmetro (Kang et al., 2008), componentes conexos (Zhu e Ghahramani, 2002), e caminhadas aleatórias (Kyrola, 2013). De qualquer maneira, a ideia é que tal processamento é aderente ao processamento sobre disco mediante acesso sequencial aos dados. Não somente isso, torna-se possível a quebra do grafo em subconjuntos
Capítulo 2. Compreensão de dados por meio de grafos

de vértices, os quais podem ser processados em paralelo, com cada parte sendo processada em uma máquina de um cluster computacional, ou em uma thread de uma máquina com múltiplos núcleos. Obtêm-se assim, algoritmos com potencial de alto desempenho, especialmente devido ao fato de que muitos algoritmos poderem ser executados de maneira assíncrona, isto é, nem todos os vértices precisam estar na mesma iteração para que se obtenha a computação correta. Esta linha de pesquisa é recente, tendo seus primeiros trabalhos publicados em 2010, com o arcabouço Pregel (Malewicz et al., 2010). Seus principais desafios são a proposição de novos algoritmos, e o aperfeiçoamento do processamento paralelo assíncrono centrado em vértice.

Detecção de comportamento lockstep

Formalmente, a ideia de comportamento lockstep é definida a seguir.

Definição 2.1: [Lockstep] Um conjunto de produtos $P$ e um conjunto de usuários $U$ determinam um lockstep (núcleo bipartido) temporalmente coerente $[n,m,\Delta t]$ se existir $P_i \subseteq P$ para todo $i \in U$ tal que:

\begin{align}
|P| &\geq m \\
|U| &\geq n \\
|P_i| &\geq |P| \forall i \in U \\
(i,j) &\in E, i \in U, j \in P_i \\
\exists t_j \in \mathbb{R} \text{ s.t. } |t_j - L_{i,j}| &\leq \Delta t, \forall i \in U, j \in P_i
\end{align}

onde $t_j$ é um momento do tempo, e $L_{i,j}$ é o momento quando a aresta $(i,j)$ foi criada. Em outras palavras, tem-se um lockstep suspeito se for encontrado um conjunto de produtos $P$ que foi recomendado por um conjunto de usuários $U$ dentro de uma janela de tempo $\Delta t$. Dada esta definição, considera-se o peso das arestas (recomendações) para se definir os conceitos de difamação – equação 2.6, e de promoção ilegítima – equação 2.7:

\begin{align}
W_{i,j} &\leq \kappa, i \in U, j \in P_i \\
W_{i,j} &\geq \kappa, i \in U, j \in P_i
\end{align}

Finalmente, considera-se o problema como um problema de otimização, no qual o objetivo é encontrar o maior número de usuários suspeitos. A equação 2.8 descreve a função objetivo, segundo a qual deseja-se encontrar $U[c]$ e $P[c]$ de maneira a maximizar o número de usuários e suas respectivas interações segundo um possível lockstep $c$.

\begin{align}
\arg \max_{U[c], P[c]} \sum_i q(L_{i,s}[c, W_{i,s}[c, P[c])
\end{align}
onde:

\[ q(u, w, P[c]) = \begin{cases} \sigma & \text{se } \sigma = \sum_{j \in P} I_{i,j} \phi(\nu_j, u_j) \lambda(w_j) \geq \rho m \\ 0 & \text{do contrário} \end{cases} \]

\[ \phi(t_{\nu}, t_u) = \begin{cases} 1 & \text{se } |t_{\nu} - t_u| \leq \Delta t \\ 0 & \text{do contrário} \end{cases} \]

\[ \lambda(g_j) = \begin{cases} 1 & \text{se } g_j \geq \kappa \\ 0 & \text{do contrário} \end{cases} \]

para promoção

\[ \lambda(g_j) = \begin{cases} 1 & \text{se } g_j \leq \kappa \\ 0 & \text{do contrário} \end{cases} \]

para difamação

As equações 2.11 e 2.12 definem os conceitos de promoção ilegítima e difamação, ao passo que a equação 2.9 mostra como os pesos das arestas são incorporados ao problema geral, considerando um limiar \( \lambda \) de aceitação da recomendação.

Definidos estes conceitos, a técnica ORFEL busca uma solução aproximada para o problema procurando pelo maior número de soluções máximas locais, ao invés de tentar encontrar máximos globais. Isto é feito “espalhando-se” sementes em vários locais ao longo do espaço de busca; uma semente se refere a um usuário escolhido aleatoriamente, e um máximo local se refere a um conjunto de usuários e de produtos suspeitos nas redondezas deste usuário escolhido aleatoriamente. A técnica ORFEL usa de processamento paralelo assíncrono para mover todas as sementes iterativamente em busca de locksteps; a cada iteração, todas as sementes movem-se um passo, sendo que cada conjunto de sementes é manipulado em uma thread separadamente. Foram realizados experimentos com 8 threads, sendo que a técnica escala automaticamente perante maior número de núcleos de processamento.

Os resultados demonstraram eficácia semelhante ao de trabalhos anteriores com eficiência notadamente superior – consulte o Apêndice A. Além da contribuição sobre o problema específico descrito, a proposta introduziu uma nova abordagem dada pelo processamento paralelo de múltiplas sementes que evoluem assincronamente dentro de um espaço de busca, o que pode ser usado em outros problemas. A técnica ORFEL foi instanciada sobre a plataforma GraphChi; durante a experimentação, todavia, observou-se que este arcabouço exigia pré-processamento, e que seu desempenho poderia ser melhorado, fatos que motivaram o trabalho M-Flash, descrito na próxima seção.

2.3 Processamento massivo de grafos

Em se tratando de grafos com alta cardinalidade, um dos problemas que surgem é a escalabilidade de processamento. Muitos trabalhos resolvem este problema por meio de clusters computacionais (Gonzalez et al., 2012)(Malewicz et al., 2010), custosos e complexos para configurar e manter. Alternativamente, uma linha de pesquisa recente (Kyrola et al., 2012)(Lin et al., 2014)(Zheng et al., 2015) objetiva escalabilidade no processamento de grafos contando com apenas um nó de processamento (um único computador). O trabalho desenvolvido pelo candidato denominado M-Flash\(^3\) (Gualdron et al., 2015b) descreve um avanço nesta direção, a qual se baseia em processamento paralelo assíncrono aplicado a algoritmos centrados em

---

\(^3\)A técnica M-Flash será submetida em fevereiro/2016; sua concepção se baseou no trabalho de Mestrado do aluno Hugo Gualdron, orientado do autor deste documento.
Figura 2.1: Esquerda: Um exemplo de matriz de adjacência (cor azul claro) segundo a técnica M-Flash com 3 intervalos lógicos ($\beta = 3$). $G^{(p,q)}$ é um bloco com as arestas no intervalo de origem $I^{(p)}$ e o intervalo de destino $I^{(q)}$. Direita: Os valores $\gamma_i$ dos vértices são armazenados como vetores, os quais são divididos nos mesmos intervalos lógicos dos blocos; por exemplo, na computação do algoritmo PageRank, são armazenados o grau de saída dos vértices e os valores intermediários do PageRank.

vértice, e que define uma otimização inovadora por meio da avaliação da densidade dos dados (arestas) do grafo em disco. A metodologia M-Flash determina um arcabouço de processamento de grafos que supera o desempenho, em tempo e em escala, dos arcabouços até então propostos para processamento de grafos baseados em um único nó. A metodologia otimiza o acesso aos dados reduzindo o número de operações de leitura e escrita ao fazer reuso dos blocos de disco mantidos em memória.

Na metodologia M-Flash, parte-se de um grafo direcionado $G = (V,E)$, com vértices $v_i \in V$, $1 \leq i \leq |V|$. Cada vértice possui um conjunto de atributos $\gamma_i = \{\gamma_{i,1}, \gamma_{i,2}, \ldots, \gamma_{i,K}\}$ armazenados separadamente como vetores em disco – ver a Figura 2.1; sendo $\gamma_{i,j}$ o $j$-ésimo atributo relacionado ao vértice $i$. Tanto os vértices (vetores) quanto as arestas (blocos) são armazenados no disco, sendo transferidos para a memória para o devido processamento.

**Organização de blocos**

Dado um grafo, os vértices são divididos em $\beta$ intervalos $I^{(p)}$, onde $1 \leq p \leq \beta$. Note que $I^{(p)} \cap I^{(p')} = \emptyset$ para $p \neq p'$, enquanto $\bigcup_{p=1}^{\beta} I^{(p)} = V$. Assim, como é mostrado na Figura 2.1, as arestas do grafo são divididas em $\beta^2$ blocos. Cada bloco $G^{(p,q)}$ possui um intervalo de origem $p$, e um intervalo de destino $q$, em que $1 \leq p,q \leq \beta$. Na Figura 2.1, por exemplo, $G^{(2,1)}$ é o bloco que contém as arestas com vértices de origem no intervalo $I^{(2)}$ e vértices de destino no intervalo $I^{(1)}$. O processo da criação dos $\beta^2$ blocos lógicos (não físicos) é chamado de particionamento do grafo. O valor de $\beta$ pode ser definido baseando-se na memória principal disponível, no número total de vértices do grafo e no tamanho dos dados que precisam ser armazenados para cada vértice. Por exemplo, para 4 GB de memória, um grafo com 2 bilhões de vértices, e 8 bytes de dados por vértice, $\beta = (2 \times 10^9 \times 8) / (4 \times 2^{30}) = 3.72$, requerendo assim $4^2 = 16$ blocos.

**Modelo de processamento M-Flash**

A técnica M-Flash se baseia no fato de que quando um grafo é organizado em disco, alguns blocos serão mais densos do que outros, de acordo com o número de arestas dos vértices de cada bloco. Verificou-se que, em trabalhos correlatos – detalhes no Apêndice B, este fato não foi levado em consideração, o que
2.3. Processamento massivo de grafos

Figura 2.2: Exemplo de ordem da computação para um grafo com três intervalos. Os blocos são processados em zig-zag (sequência de setas vermelha, laranja, e amarela) permitindo o reuso dos dados dos vértices. No exemplo, faz-se reuso dos vértices de destino ao longo da coluna de destino 1, e faz-se reuso dos vértices de origem da linha 3 quando se passa do bloco \( G^{(3,1)} \) para o bloco \( G^{(3,2)} \).

Figura 2.3: Exemplo de operações de I/O para processar o bloco denso \( G^{(2,1)} \).

prejudica o desempenho. Assim, além de propor um inovador mecanismo de escalonamento dos dados em disco, a técnica M-Flash também faz uso de processamentos distintos definidos dinamicamente segundo a densidade das arestas em um dado bloco. Definem-se, portanto, dois tipos de processamento, para blocos densos e para blocos esparsos.

**Processamento de blocos densos (PBD):** após dividir logicamente o grafo em blocos, estes são processados em zig-zag, como ilustrado na Figura 2.2. Durante a leitura, após carregar na memória um intervalo de destino (por exemplo \( I^{(1)} \)), todas as arestas (blocos) cujos vértices de origem têm como destino este intervalo são processadas. Isto pois, o processamento centrado em vértices funciona propagando dados de um dado vértice para seus vizinhos (destinos). Este esquema reduz o número de escritas em disco, uma vez que todo o processamento do bloco de destino será feito em memória, sendo reescrito em disco apenas após o completo processamento da respectiva linha – ver a seta vertical vermelha da Figura 2.2. Uma otimização adicional ocorre quando há a troca de colunas na leitura em zig-zag; a ordem de processamento dos blocos aproveita os valores relacionados aos vértices de origem disponíveis na memória – ver a seta laranja da Figura 2.2.
Capítulo 2. Compreensão de dados por meio de grafos

**Processamento de blocos esparsos** \(^4\) (PBE): o desempenho do processamento de blocos densos cai quando os blocos são esparsos; isto pois o modelo precisará ler mais dados dos vértices (vetores) de origem do que os dados contidos nos blocos esparsos (arestas). Para resolver isso, o processamento **PBE** baseia-se em duas etapas (ver Figura 2.3). Na primeira etapa, o método lê os vértices e arestas reordenando os dados, os quais dão origem a blocos temporários contendo dados tanto dos vértices quanto das arestas. Na segunda etapa, estes dados são lidos e processados sequencialmente, e os resultados da computação dão origem a versões atualizadas dos intervalos de vértices de destino. No entanto, esta abordagem não apresenta bom desempenho para blocos densos, o que impede que um único tipo de processamento seja usado.

![Figura 2.4: Etapas do processamento de blocos esparsos.](image)

Qual processamento usar, denso ou esparsó?
The método **PBD** é bom para blocos densos enquanto o método **PBE** é a melhor opção para blocos esparsos. Mas, como decidir se um bloco é denso ou esparsó quando não temos controle sobre a ordem das arestas? Para responder essa questão foi feita uma comparação do custo de I/O entre os modelos **PBD** (Equação 2.13) e **PBE** (Equação 2.14) baseada no modelo teórico de custo de I/O proposto por Aggarwal e Vitter (Aggarwal e Vitter, 1988). A complexidade de cada um dos algoritmos foi parametrizada tendo em consideração que são transferidos blocos de dados de tamanho \(B\) entre a memória e o disco. Após o cálculo do custo dos modelos, são usadas as equações 2.15 e 2.16 para selecionar o melhor modelo para cada bloco. Nas equações, \(\xi\) é o número de arestas em \(G^{(p,q)}\), \(\vartheta\) é o número de vértices no intervalo, \(\phi\) e \(\psi\) são, respectivamente, as quantidades de bytes para representar um vértice e uma aresta e. A partir da classificação, pode-se decidir, em tempo de execução, qual o melhor processamento a ser usado, aproveitando do melhor desempenho oferecido por cada tipo de processamento.

\[
\Theta \left( NBP \left( G^{(p,q)} \right) \right) = \Theta \left( \frac{(2\vartheta \phi + \xi \psi)}{B} \right) \tag{2.13}
\]

\[
\Theta \left( EBP \left( G^{(p,q)} \right) \right) = \Theta \left( \frac{(\xi \psi + \vartheta \phi / \beta + 2\xi \phi \psi + \vartheta \psi)}{B} \right) \tag{2.14}
\]

\[
\Theta \left( \frac{EBP}{NBP} \right) = \Theta \left( \frac{1}{\beta} + \frac{2\xi \phi}{\vartheta} \right) \tag{2.15}
\]

\(^4\)Adaptação livre da denominação em inglês *Streaming Partition Processing (SPP).*
### Algoritmo 2.3 \textit{MAlgorithm}: Interface de programação de algoritmos em M-Flash.

<table>
<thead>
<tr>
<th>Ação</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{initialize} (Vertex ( v ))</td>
<td>Carrega o valor de início dos vértices de destino;</td>
</tr>
<tr>
<td>\textit{gather} (Vertex ( u ), Vertex ( v ), EdgeData data)</td>
<td>A operação \textit{gather} coleta dados a partir dos vértices vizinhos;</td>
</tr>
<tr>
<td>\textit{sum} (Accum ( v_{-1} ), Accum ( v_{-2} ), Accum ( v_{\text{out}} ))</td>
<td>A operação \textit{sum} sumariza os dados coletados por diferentes threads para o mesmo vértice;</td>
</tr>
<tr>
<td>\textit{apply} (Vertex ( v ))</td>
<td>A operação \textit{apply} atualiza e armazena em disco o novo dado gerado para o vértice,</td>
</tr>
</tbody>
</table>

\[
\text{TipoDoBloco} \left( G^{(p,q)} \right) = \begin{cases} 
\text{sparse,} & \Theta \left( \frac{E_{BP}}{N_{BP}} \right) < 1 \\
\text{dense,} & \text{otherwise} 
\end{cases}
\] (2.16)

É importante notar que toda esta complexidade poderia ser evitada ao se reordenar os dados do grafo em disco antes do processamento. Isto poderia ser feito por meio de algoritmos de agrupamento como CrossAssociation (Chakrabarti et al., 2004), Metis (Karypis e Kumar, 1998) ou Slash-Burn (Lim et al., 2014). Conceitualmente, esses algoritmos trocam a ordem das linhas e colunas da matriz de adjacência do grafo, o que leva a ter alguns blocos do grafo muito densos e outros muito esparsos segundo a distribuição de potência do grau nos grafos reais (Faloutsos et al., 1999). No entanto, os problemas tratados aqui se referem a grafos que podem ocupar um disco inteiro (centenas de GBs), de modo que a reordenação pode demandar um tempo de processamento inviável. Assim, um dos objetivos do trabalho foi evitar a necessidade de reordenação, como realizado em outros trabalhos (Kyrola et al., 2012)(Han et al., 2013), os quais dependem de uma disposição contígua lógica e física dos dados; assim, o trabalho proposto alcança resultados mais escaláveis por meio de seu modelo de processamento dinâmico.

**Modelo de programação**

A técnica M-Flash tem suporte para grafos direcionados com dados nas arestas por meio da interface para programação de algoritmos \textit{MAlgorithm} (algoritmo 2.3). A interface \textit{MAlgorithm} é um modelo centrado em vértice que armazena os resultados da computação nos vértices de destino. A interface foi inspirada em modelos de programação anteriores como o GIM-V do Pegasus (Kang et al., 2012) e GAS do PowerGraph (Gonzalez et al., 2012). A interface \textit{MAlgorithm} poder ser empregada na implementação dos algoritmos mais comuns de grafos, em especial daqueles que são iterativos como PageRank, passeios aleatórios, componentes conexos, e cálculo aproximado do diâmetro.

A interface \textit{MAlgorithm} possui quatro operações: \textit{initialize}, \textit{gather}, \textit{sum}, e \textit{apply}. A operação \textit{initialize} carrega o valor de início dos vértices de destino; a operação \textit{gather} coleta dados a partir dos vértices vizinhos; a operação \textit{sum} sumariza os dados coletados por diferentes threads para o mesmo vértice; e finalmente, a operação \textit{apply} atualiza e armazena em disco o novo dado gerado para o vértice, sendo disponibilizado para iterações futuras. Esta interface foi definida seguindo o modelo mais usado da literatura, permitindo que algoritmos existentes possam ser executados com poucas alterações.

Como apresentado no Apêndice B, a técnica M-Flash passou por ampla experimentação, sendo comparada com as técnicas mais recentes da literatura segundo consumo de memória, acessos a disco, tempo de processamento, e análise teórica. A técnica se mostrou mais eficiente em todos os aspectos, sobretudo quando o grafo não cabe em memória, situação quando outras técnicas degradam rapidamente.
2.4 Fatoração matricial aplicada sobre a base de dados DBLP

Uma das possibilidades de se trabalhar com grafos é o cálculo de propriedades algébricas, as quais se convertem em informação semântica quando interpretadas à luz do domínio de dados. Em um dos trabalhos desenvolvidos (Quille et al., 2014)\textsuperscript{5}, o autor usou de Decomposição por Valor Singular (do inglês, *Singular Value Decomposition* (SVD)) sobre a base de dados DBLP (*Digital Bibliography & Library Project*) (Ley, 2009), o maior repositório de dados de publicações em Ciência da Computação. O modelo de dados da DBLP está esquematizado na Figura 2.5 segundo o modelo entidade-relacionamento (mapeável para o Modelo Relacional de dados); a cardinalidade dos dados é apresentada na Tabela 2.1. Este domínio define um conjunto de dados ricamente constituído de informações semânticas, levantando diversas questões analíticas; sobre este domínio, a SVD foi usada para a detecção de comunidades.

![Figura 2.5: Modelo entidade-relacionamento da DBLP.](image)

Tabela 2.1: Instâncias das entidades envolvidas na análise da DBLP – dados de 2012.

<table>
<thead>
<tr>
<th>Entidades</th>
<th>Número de entidades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autores</td>
<td>1.054.199</td>
</tr>
<tr>
<td>Artigos</td>
<td>1.801.576</td>
</tr>
<tr>
<td>Eventos</td>
<td>3.050</td>
</tr>
<tr>
<td>Veículos</td>
<td>4.262</td>
</tr>
</tbody>
</table>

Decomposição por Valor Singular (SVD)

O processo SVD é uma modalidade de fatoração matricial, o que o torna aplicável a dados representados como grafos, pois estes são naturalmente representados como matrizes de adjacência. Assim, dada uma matriz $A_{n \times m}$, tal que $A^2$ é positiva-semidefinida, a Decomposição por Valor Singular pode ser expressa segundo a igualdade:

$$A_{n \times m} = U_{n \times r} S_{r \times r} V_{r \times m}^T = \sum_{i=1}^{\text{rank}(A)} (\lambda_i u_i \otimes v_i) \quad (2.17)$$

Onde:

- o $\text{rank}(A)$, $\text{rank}(A) \leq \min\{n, m\}$, se refere ao número de linhas (ou colunas) linearmente independentes da matriz $A$;
- o símbolo $\otimes$ é o produto externo dos dois vetores de dados;
- a matriz $U_{n \times r}$ é constituída de $n$ vetores de “entrada” $u_i$, $1 \leq i \leq n$;

\textsuperscript{5}Este trabalho foi publicado nos *Proceedings of the ACM Symposium on Applied Computing* (Qualis A1), em 2014; sua concepção se baseou no trabalho de Mestrado da aluna Rosa Quille, orientada do autor deste documento.
2.4. Fatoração matricial aplicada sobre a base de dados DBLP

- a matriz diagonal \( S_{r \times r} \) é constituída por \( r \) valores singulares \( s_{i,j} \); \( i = j, 1 \leq i, j \leq r \);
- a matriz \( V_{r \times m}^T \) é constituída de \( r \) vetores de “saída” \( v_i \), \( 1 \leq i \leq r \).

Os vetores da matriz \( V^T \) são ortonormais, ou seja, linearmente independentes e com norma unitária. Eles determinam uma nova base capaz de representar \( A \) por meio de combinações lineares de \( US \).

SVD aplicada a dados relacionais

No caso de \( A \) ser um grafo representado como uma matriz de adjacências definida a partir de um Banco de Dados Relacional, o principal resultado da SVD é o agrupamento dos \( m \) elementos que determinam as linhas da matriz \( A \) em função de como estes se relacionam com os \( n \) elementos que determinam as colunas da matriz. Os agrupamentos são indicados pelos maiores valores singulares da fatoração SVD como ilustrado nas Figuras 2.6 e 2.7. Em um grafo de co-autoria, por exemplo, esta aplicação poderia identificar quais são as equipes mais atuantes de autores em função de quais artigos eles publicam juntos. Alternativamente, como no trabalho de Nan Du e Faloutsos (Du et al., 2010), a SVD pode ser aplicada ao longo do tempo para identificar, por exemplo, quais são os orientadores mais atuantes na instituição e como eles têm se comportado. É interessante observar que os agrupamentos detectados via SVD não são da mesma natureza que o agrupamento categórico tradicionalmente desenvolvido em mineração de dados, mas sim um agrupamento baseado no relacionamento entre diferentes entidades, assim como no trabalho de Prakash et al. (Prakash et al., 2009).

**Figura 2.6:** Grafo exemplo - relacionamento “professor participa de evento” muitos-para-muitos entre instâncias de professores e instâncias de eventos. Os pesos das arestas indicam o número de participações de um dado professor em um dado evento.

Mineração de dados

Para o processamento adequado dos dados da DBLP, foi necessário um processo de mineração de dados que solucionasse os problemas de cardinalidade, dados espúrios, e não conformidade da informação textual. O processo, ilustrado no diagrama da Figura 2.8, teve sete passos: Pré-processamento e limpeza de dados, Modelagem de dados, Transformação Relacional, Seleção de dados, Processamento SVD, Análise de dados, e Avaliação e interpretação.

No pré-processamento, a informação textual passou por etapas de conversão para código ASCII, identificação de termos isolados (tokenization), remoção de palavras auxiliares (stopwords), e radicalização (stemming). Os dados foram, então, modelados e transferidos para um Banco de Dados Relacional para que pudessem ser manipulados mais facilmente, incluindo operações de seleção e agregação. Nesta etapa, foram executadas operações de filtragem para a resolução de problemas inerentes aos dados.
Capítulo 2. Compreensão de dados por meio de grafos

**Autores x Eventos**

\[
A_{mxn} = \begin{bmatrix}
\begin{array}{cccccc}
2 & 3 & 5 & 0 & 1 & 0 \\
4 & 7 & 5 & 0 & 0 & 1 \\
6 & 15 & 7 & 2 & 0 & 0 \\
8 & 4 & 8 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & 2 & 9 \\
1 & 0 & 0 & 3 & 1 & 6 \\
5 & 2 & 0 & 2 & 6 & 5
\end{array}
\end{bmatrix} \begin{bmatrix}
0.2428 & 0.0280 & 0.1447 & 0.2416 & 0.8505 & -0.2411 \\
0.4040 & 0.0093 & -0.0242 & -0.5247 & -0.0782 & -0.2223 \\
0.7353 & 0.0289 & -0.5276 & -0.5226 & -0.2223 & -0.1750 \\
0.4760 & 0.1084 & 0.7738 & 0.1737 & -0.3209 & -0.3180 \\
0.0447 & -0.8250 & 0.1199 & -0.2591 & 0.1569 & -0.1799 \\
0.0432 & -0.4377 & 0.1469 & -0.1460 & -0.1324 & -0.0867 \\
0.0817 & -0.3380 & -0.2679 & 0.9346 & -0.2683 & 0.0706 \\
\end{bmatrix}
\]

\[
S_{rxr} = \begin{bmatrix}
23.4465 & 0 & 0 & 0 & 0 & 0 \\
13.1187 & 0 & 0 & 0 & 0 & 0 \\
0 & 7.4292 & 0 & 0 & 0 & 0 \\
0 & 0 & 5.1723 & 0 & 0 & 0 \\
0 & 0 & 0 & 2.2571 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.0596 & 0
\end{bmatrix}
\]

\[
V_{rxn}^T = \begin{bmatrix}
0.4624 & 0.7103 & 0.5199 & 0.0867 & 0.0369 & 0.0506 \\
0.0614 & 0.0250 & 0.0958 & -0.5247 & -0.3116 & -0.7837 \\
0.5619 & -0.6729 & 0.4229 & -0.5247 & -0.3116 & -0.7837 \\
0.0491 & -0.4377 & 0.1469 & -0.1460 & -0.1324 & -0.0867 \\
-0.6761 & -0.0637 & 0.7019 & -0.1480 & -0.1460 & -0.0867 \\
0.0849 & 0.1382 & -0.8214 & 0.9346 & -0.2683 & 0.0706 \\
\end{bmatrix}
\]

**Figura 2.7:** Decomposição por Valor Singular correspondente ao grafo da Figura 2.6.

**Figura 2.8:** Metodologia de mineração de dados aplicada sobre a DBLP.

Como exemplo dos problemas solucionados no processo de mineração está o fato de que a DBLP é fortemente desequilibrada em relação à produção dos autores em número de artigos. Para verificar este aspecto, ilustra-se a distribuição dos autores-artigos na Figura 2.9(a). O gráfico mostra uma distribuição de cauda longa em que a maioria dos autores não tem mais que 22 artigos – mais precisamente, essa porção corresponde a 1.016.354 autores, ou ∼96% das instâncias. No gráfico, pode-se observar uma distribuição de lei de potência de acordo com a equação:

\[
y \propto 10^6 x^{-1.06}
\]  
(2.18)
onde \( y \) se refere ao número de autores que possuem \( x \) artigos.

Observa-se uma instância do Princípio de Pareto, ou a regra 80-20, na qual \( \sim 24\% \) dos autores são responsáveis por \( \sim 76\% \) dos artigos – ver Figura 2.9(b). Esta observação permitiu restringir a análise aos \( \sim 24\% \) autores mais prolíficos, ou seja, os 255.455 autores com 4 ou mais artigos. Em uma outra análise, foi possível selecionar quais termos dos títulos dos artigos publicados usar para análise a fim de se entender a relação autores \( \times \) temos usados nos títulos das publicações. Para seleção dos termos, baseou-se em frequência e distribuição segundo a Lei de Zipf (Luhn, 1958), a qual afirma que os termos mais significativos são aqueles que não são muito comuns nem aqueles que são muito raros. Segundo esta teoria, a relevância dos termos é dada pela curva Gaussiana centrada na média \( \mu \) com desvio padrão \( \sigma \) da distribuição de uso dos termos, dada pela equação 2.19:

\[
TS(i) = f(t_i; \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(t_i-\mu)^2/2\sigma^2}
\]

onde \( t_i \) se refere ao i-ésimo termo.

Esta teoria permitiu reduzir em \( \sim 90\% \) o número de termos usados na análise. A redução de autores e de termos produziu um conjunto mais denso e significativo, evitando autores esporádicos e termos insignificantes. Além disso, a análise SVD pôde ser executada em menos tempo devido à redução de cardinalidade, ao mesmo tempo em que se produziram resultados potencialmente mais significativos.

![Figura 2.9: Distribuição autores-artigos. (a) número de artigos \( \times \) número de autores. (b) Rank-plot sobre o índice do autor \( \times \) número de artigos.](image)

**Resultados**

Os resultados da mineração de dados baseada em SVD são reportados em detalhes no Apêndice C. Estes resultados demonstram o processo completo sobre um caso real altamente semântico, seus desafios e soluções propostas. Foram analisados não apenas Autores \( \times \) Termos, mas também Termos \( \times \) Veículos de publicação. Na Tabela 2.2, por exemplo, observam-se algumas das principais comunidades em Ciência da Computação identificadas via análise SVD sobre a relação Autores \( \times \) Termos. Já, na Tabela 2.3, é possível verificar as comunidades mais atuantes de acordo com os termos mais usados nos principais veículos de publicação.

Além de definir um processo detalhado de mineração de dados com pré-processamento de texto, este trabalho demonstrou que o uso de Decomposição por Valor Singular pode detectar padrões interessantes
em situações onde conjuntos disjuntos interagem entre si. O trabalho mostrou como a decomposição pode ser aplicada a diversos relacionamentos encontrados em Bancos de Dados Relacionais, onde a interação entre conjuntos disjuntos é frequente. No processo, foi possível notar que o processamento de texto pode impulsionar os resultados da análise, produzindo resultados semanticamente mais densos e em menos tempo. Como reportado no Apêndice C, os resultados da SVD podem ser interpretados considerando seus três elementos U, S, e T – veja a Equação 2.17. No caso da interação autor-evento, por exemplo, além da identificação das comunidades de autores definidas por suas participações em eventos, torna-se possível identificar os autores mais ativos nos eventos, e também os eventos mais representativos em relação à participação de autores. O trabalho, portanto, mostrou evidências de que uso da SVD em bases relacionais é adequado e promissor.

### 2.5 Técnicas de análise de grafos aplicadas sobre a DBLP

Na mesma linha do trabalho da Seção 2.4, dois outros trabalhos (Gimenes et al., 2015)(Gimenes et al., 2014) exploraram as possibilidades de computação de métricas a partir da base de dados da DBLP representada como um grafo. Ambos os trabalhos descrevem processos que usam medidas calculadas a partir de vértices, de pares de vértices, ou de todo o grafo (métricas globais) para caracterizar o domínio de dados. Dada a semântica da DBLP, as medidas são interpretadas semanticamente ao passo que...
2.5. Técnicas de análise de grafos aplicadas sobre a DBLP

se descreve o processo de maneira generalizável a outros domínios. Nestes trabalhos foram usadas as seguintes medidas?:

- **weakly-connected components (WCC)**: a distribuição de componentes fracamente conectadas se refere a um subgrafo não direcionado no qual todo nó possui um caminho para todo outro nó – nesta distribuição, conta-se o tamanho de todas as WCCs;

- **average clustering coefficient (ACC)**: o coeficiente médio de agrupamento indica a tendência de um conjunto de vértices de mesmo grau de participar de um agrupamento, isto é, uma comunidade dentro da qual os vértices são mais intensamente interconectados;

- **degree distribution**: a distribuição do grau dos vértices indica quão intensamente os autores interagem uns com os outros, revelando padrões sobre o processo de crescimento do grafo;

- **effective diameter evolution**: a evolução do diâmetro efetivo do grafo indica quão intenso é o fenômeno de “small world” ao longo do tempo;

- **co-authoring predictability**: neste contexto, em particular, refere-se ao quão previsível são as novas conexões, ou co-autorias entre os autores de acordo com as propriedades observadas no passado do grafo.

O conjunto destes trabalhos propôs uma metodologia que coloca juntas abordagens estatísticas, topológicas, algorítmicas, e algébricas. Sobre o domínio da DBLP, estas medidas proveram ampla interpretação, o que deu origem a hipóteses sobre o comportamento dos autores. Como apresentado nos Apêndices D e E, esta é uma área de pesquisa de amplo interesse, tendo produzido trabalhos que procuram verificar se autores que participam do mesmo evento tendem a ser mais colaborativos (Ośieck et al., 2009), ou se autores com mesmo perfil tendem a se relacionar (Aiello et al., 2012). Contribuições desta linha de trabalho podem ser usadas em outros contextos; especificamente, o presente trabalho identificou alguns padrões que levaram a hipóteses de interesse para o domínio da DBLP, como detalhado a seguir.

**Weakly-connected components (WCC)**
A Figura 2.10 mostra a distribuição WCC sobre a DBLP considerando co-autoria de trabalhos. Na figura, vê-se que 13% dos autores formam pequenas comunidades de até 30 autores, ao passo que uma componente gigante formada por 87% dos vértices contém autores de todo o mundo relacionados diretamente ou indiretamente. Mais interessante é a presença de ~44.000 sub comunidades formadas por ~120.000 autores. Este fato indica que há produção científica de autores isolados que produziram algum tipo de trabalho apenas, possivelmente, para a obtenção de seus títulos, não dando continuidade à pesquisa, ou não se integrando à grande comunidade produtora de conhecimento. Estes autores são, potencialmente, indicativos de veículos/events de menor qualidade, pois este padrão de produção indica trabalhos desconectados das tendências do estado da arte, as quais tendem a se disseminar ao longo de toda a rede.

**Average Clustering Coefficient (ACC)**
Por meio do gráfico de grau de nó × ACC – ver Figura 2.11, pode-se verificar que valores elevados (perto
Capítulo 2. Compreensão de dados por meio de grafos

de 1) de ACC são observados somente para nós com grau até 10, com os valores decrescendo ao longo da sequência segundo a lei de potência: $ACC \propto grau^{-1.06}$. Isso significa que nós com grau até 10 tendem a formar agrupamentos nos quais todos os vértices são altamente interligados, tendência que diminui com o aumento do grau.

A hipótese aqui é a de que os autores tendem a colaborar com co-autores de seus co-autores, formando triângulos na rede. Outra explicação vem do fato de que grafos hierarquicamente organizadas tendem a apresentar diminuição progressiva de ACC (Ravasz e Barabasi, 2002). Além disso, os autores mais antigos (orientadores) tendem a ter grau e redes de colaboração cada vez maiores, o que significa que eles são menos propensos a ser parte de um conjunto bem definido e altamente interligado. Em vez disso, eles tendem a ser ligados a diversos outros sub grafos que tendem a ser esparsos com o aumento do número de vértices.

**Degree Distribution**

A partir do gráfico de distribuição de grau – Figura 2.12, pode-se ver que a distribuição de graus da DBLP está obedecendo a uma lei de potência com expoente $\gamma \approx -1.36$; esse fato indica que quanto maior o grau (mais co-autorias), menor o número de autores. De acordo com Leskovec et al. (Leskovec et al., 2007), há uma relação entre a distribuição de graus e um processo denominado densificação, segundo o qual quanto mais nós entrarem, mais arestas tendem a aparecer de acordo com a lei de potência a seguir:

$$e(t) \propto n(t)^a$$  \hspace{1cm} (2.20)

onde $e(t)$ é o número de arestas e $n(t)$ é o número de vértices para um dado momento $t$. Note que o gráfico da Figura 2.12 não descreve densificação como descrito pela equação 2.20. Mas, segundo Leskovec, uma distribuição de grau com lei de potência de expoente $\gamma$ indica uma densificação com expoente $a = 2/|\gamma|$. 
2.5. Técnicas de análise de grafos aplicadas sobre a DBLP

Para a DBLP, portanto, teria-se uma densificação $e(t) \propto n(t)^{2/1.36} = e(t) \propto n(t)^{1.47}$. Esta densificação é compreensível, pois mesmo que a entrada de novos autores, em um primeiro momento, torne a rede mais esparsa, aos longo de décadas, os autores irão ter inúmeros co-autores, aumentando a densidade.

Pode-se observar densificação semelhante em outros ambientes, como a web. No entanto, pode-se perguntar por que o número de arestas cresce exponencialmente em um ambiente onde novas arestas não são tão baratas como na web; mas que, em vez disso, dependem de laboriosas publicações científicas. A hipótese é a de que os títulos de Mestre e Doutor, certificações de conhecimento e experiência, eram originamente concedidos *ad hoc* somente quando um pesquisador alcançava produção suficiente. Nas últimas décadas, no entanto, estes títulos se tornaram cursos regulares com cronogramas bem definidos e produção esperada. Consequentemente, uma demanda por “onde publicar”, ao invés de “o que publicar”, foi criada. Este fato levou a uma literatura científica prolixa e que apresenta diferentes graus de qualidade.

**Effective diameter**

Ao se inspecionar a evolução efetiva do diâmetro da rede de co-edição (autores que editaram o mesmo periódico) – ver a Figura 2.13, é possível ver que o diâmetro efetivo começa a encolher depois de um certo ponto no tempo – em torno do ano 1995. Esse fato sugere que, antes deste ano, surgiam novos periódicos com novas oportunidades de edição até que se alcançou um pico, quando os mesmos editores começaram a editar veículos já existentes. O padrão da curva também sugere que uma quantidade grande de arestas começou a entrar na rede neste momento, iniciando um período de densificação, e consequente diminuição do diâmetro.

Uma possível explicação é que as comissões de editores tendem a ter os mesmos membros que se alternam entre um conjunto limitado de periódicos, ano após ano, com pouca entrada de novos editores. Como tal, a distância entre quaisquer dois editores tende a diminuir ao longo do tempo. Possivelmente, este é o caso porque a edição de publicações é uma tarefa que exige experiência e conhecimento, características de um conjunto limitado de pesquisadores. Além disso, como o número de editores
é limitado, a atividade de edição é uma questão de disputa na comunidade científica, o que levanta obstáculos adicionais para autores recém-chegados.

Co-authoring predictability

Também é relevante avaliar a previsibilidade das novas co-autorias na DBLP. Para tanto, usou-se de técnicas de predição de arestas baseadas em métricas extraídas do grafo. Especificamente, para cada autor (ou par de autores) foram extraídas as seguintes métricas: número de autores comuns (a), coeficiente de Jaccard (b), interconexão preferencial (c), coeficiente de Adamic-Adar (d), índice de alocação de recursos (f), e caminho local (g) – mais detalhes no trabalho de Gimenes et al. (Gimenes et al., 2014). Organizadas como vetores, estas métricas são então utilizadas com os seguintes classificadores de aprendizado de máquina supervisionado: J48, Naïve Bayes (NB), Multilayer Perceptron (MLP), Bagging, e Random Forest (RF) – detalhes no trabalho de Witten et al. (Witten et al., 2011). Para a predição de links, usa-se como treino as arestas existentes no passado da rede, anos de 1995 a 2005; e, como teste de acurácia, os anos de 2006 a 2007. A acurácia, aqui, indica a previsibilidade das novas co-autorias da DBLP, isto pois usou-se de algoritmos já consagrados com eficácia comprovada, de modo que se a acurácia for baixa, o desafio é o conjunto de dados. Foram medidos Precisão-Recuperação, F-Measure, e Área Under the Curve (AUC) para curvas Receiver Operating Characteristic (ROC). Também se considerou diferentes perfis de autores de acordo com seu grau variando entre maior que 1, até maior que 8. O resultado é apresentado na Tabela 2.4; quanto maior o valor, maior a previsibilidade. Verificou-se que a DBLP tem alta previsibilidade, perto de 90% de acerto, especialmente para autores com maior grau. A hipótese é que isto se deve ao modelo de interação entre os autores, majoritariamente definido por co-autoria orientador-aluno, ex-orientador-aluno, e aluno-aluno de um mesmo orientador/ex-orientador.

Mais detalhes da metodologia são apresentados nos Apêndices D e E. Nos textos são mostrados detalhes da preparação dos dados, interpretação dos resultados, e exercícios algébricos para mais bem entender o domínio da DBLP. Além do processo e da metodologia, estes trabalhos também contribuíram com a compreensão de que múltiplas análises podem prover uma perspectiva mais ampla de um dado.
domínio. Muito embora a rica semântica da DBLP possa sugerir hipóteses por si só, este nem sempre é o caso; para domínios menos semânticos, o processo analítico é o que dá sustentação à detecção de causas e efeitos.

2.6 Sumarização visual de grafos

Nesta seção é apresentada uma técnica de sumarização de grafos instanciada sobre o sistema GMine, de autoria do candidato. O trabalho GMine (RODRIGUES-JR et al., 2006a)(RODRIGUES-JR et al., 2006b) se refere a um arcabouço de análise interativa de grafos que reúne técnicas de visualização interativa e técnicas analíticas. O trabalho de visualização foi desenvolvido durante o doutorado, finalizado em 2007, enquanto que as técnicas algorítmicas foram desenvolvidas posteriormente. Neste documento, reporta-se a contribuição, publicada em 2013, referente a uma técnica de sumarização de grafos que foi desenvolvida e agregada ao sistema8 (RODRIGUES-JR et al., 2013b). A técnica considera o problema de se resumir redes inteiras a partir de vértices de interesse apontados visualmente pelo usuário. Para resolver o problema, usa-se de processamento algébrico definido sobre teoria de processos estocásticos em combinação com um algoritmo guloso e com programação dinâmica. Os resultados demonstraram uma técnica de sumarização capaz de resumir visualmente redes complexas de acordo com os interesses do usuário.

Dado um grafo com $N$ vértices e um conjunto $Q$ de vértices de interesse, a técnica de sumarização desenvolvida objetiva encontrar o subgrafo central que mais bem resume a interconexão entre os vértices de $Q$. Tal subgrafo corresponde a um conjunto de vértices (e arestas) com cardinalidade de vértices

---

8Esta técnica foi publicada na IEEE Transactions on Knowledge and Data Engineering (Qualis A1), em 2013; sua concepção se deu, principalmente, em colaboração com o co-autor Hanghang Tong, atualmente, professor da Arizona State University, AZ, EUA.
### Tabela 2.4: Acurácia de predição de arestas dada pelas medidas de Precisão-Revocação, F-Measure, e Area Under the Curve (AUC) de cinco classificadores supervisionados sobre a base de dados da DBLP.

Testes realizados com autores de diferentes graus mínimos: \(d \geq 1, d \geq 2, d \geq 4, d \geq 6, \text{and } d \geq 8\).

<table>
<thead>
<tr>
<th>(d)</th>
<th>Classificador</th>
<th>PREC</th>
<th>REC</th>
<th>F-MEAS</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>J48</td>
<td>0,723</td>
<td>0,706</td>
<td>0,7</td>
<td>0,764</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0,741</td>
<td>0,585</td>
<td>0,505</td>
<td>0,626</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0,562</td>
<td>0,555</td>
<td>0,541</td>
<td>0,593</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0,809</td>
<td>0,8</td>
<td>0,798</td>
<td>0,887</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0,877</td>
<td>0,868</td>
<td>0,867</td>
<td>0,939</td>
</tr>
<tr>
<td>2</td>
<td>J48</td>
<td>0,787</td>
<td>0,759</td>
<td>0,753</td>
<td>0,817</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0,777</td>
<td>0,598</td>
<td>0,52</td>
<td>0,648</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0,628</td>
<td>0,618</td>
<td>0,61</td>
<td>0,639</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0,84</td>
<td>0,83</td>
<td>0,829</td>
<td>0,913</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0,914</td>
<td>0,903</td>
<td>0,902</td>
<td>0,977</td>
</tr>
<tr>
<td>4</td>
<td>J48</td>
<td>0,852</td>
<td>0,845</td>
<td>0,844</td>
<td>0,87</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0,773</td>
<td>0,585</td>
<td>0,499</td>
<td>0,704</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0,715</td>
<td>0,714</td>
<td>0,713</td>
<td>0,735</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0,846</td>
<td>0,841</td>
<td>0,841</td>
<td>0,925</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0,917</td>
<td>0,913</td>
<td>0,912</td>
<td>0,974</td>
</tr>
<tr>
<td>6</td>
<td>J48</td>
<td>0,827</td>
<td>0,771</td>
<td>0,761</td>
<td>0,79</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0,778</td>
<td>0,601</td>
<td>0,526</td>
<td>0,727</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0,695</td>
<td>0,679</td>
<td>0,672</td>
<td>0,74</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0,844</td>
<td>0,83</td>
<td>0,828</td>
<td>0,913</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0,897</td>
<td>0,888</td>
<td>0,887</td>
<td>0,972</td>
</tr>
<tr>
<td>8</td>
<td>J48</td>
<td>0,861</td>
<td>0,839</td>
<td>0,836</td>
<td>0,867</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0,786</td>
<td>0,626</td>
<td>0,566</td>
<td>0,741</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0,725</td>
<td>0,719</td>
<td>0,717</td>
<td>0,785</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0,883</td>
<td>0,866</td>
<td>0,865</td>
<td>0,94</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0,914</td>
<td>0,908</td>
<td>0,907</td>
<td>0,971</td>
</tr>
</tbody>
</table>

máxima \(b\) (fornecida pelo analista) que possuem conexões diretas ou indiretas a todos os vértices de interesse. Resolver este problema corresponde a sumarizar o grafo reduzindo-se o número de arestas para se compreender uma dada rede; a redução ocorre de \(|N|(N-1)/2\) arestas a serem consideradas (no pior caso, ilustrado nas Figuras 2.14(a) e Figura 2.14(b)) para um número proporcional à cardinalidade \(b\), possivelmente com \(b << N\), o que define um problema solucionável via um processo estocástico Markoviano cuja resolução eficiente depende de operações algébricas não triviais.

Formalmente, dados \(Q\) vértices de interesse de um grafo \(G=\{V,E\}\) – Figura 2.14(c), deseja-se encontrar o subconjunto de vértices \(CP \in V\) que irão determinar um subgrafo induzido \(G[CP]\) com cardinalidade máxima \(b\) e possuindo conexões para todos os \(Q\) vértices – Figura 2.14(d). Considerando este problema, a técnica se baseia na maximização da adequação, definida como uma métrica numérica, do subgrafo a ser identificado.

Para um dado vértice \(j\), podem-se definir dois tipos de métricas de adequação:

- Seja \(r(i,j)\) a adequação do vértice \(j\) em relação ao vértice de interesse \(q_i\);
- Seja \(r(Q,j)\) a adequação do vértice \(j\) em relação ao conjunto de vértices de interesse \(Q\).

O que permite definir a adequação de um subgrafo \(CP\) como:
2.6. Sumarização visual de grafos

\[ g(\text{CP}) = \sum_{j \in \text{vertices(\text{CP})}} r(\mathcal{Q}, j) \]  

(2.21)

Assim, surgem dois problemas para se identificar o subgrafo central: 1) como definir uma adequação \( r(\mathcal{Q}, j) \) para um dado vértice \( j \); 2) como computar um subgrafo central que maximiza a função \( g(\text{CP}) \).

Para resolver estes problemas, usa-se de um processo estocástico:

- Seja \( r_{i,j} \) a \textit{probabilidade de transição estacionária} de que uma partícula se encontrará no vértice \( j \) quando ela realizar uma caminhada aleatória com recomeço a partir de um vértice \( q_i \);

- Seja \( r(\mathcal{Q}, j, \mathcal{Q}) \) a \textit{probabilidade de encontro}, isto é, a \textit{probabilidade de transição estacionária} de que todas as \( Q \) partículas, realizando caminhada aleatória com recomeço a partir dos vértices de \( \mathcal{Q} \), se encontrarão no vértice \( j \) no estado estacionário.

Primeiro, deseja-se computar a adequação \( r(i, j) \) de um único vértice \( j \) em relação a um único vértice \( q_i \). Para tanto, usa-se caminhada aleatória com recomeço a partir de um vértice \( q_i \). Suponha uma partícula aleatória iniciando em \( q_i \); esta partícula é transmitida iterativamente para sua vizinhança com probabilidade proporcional ao peso da aresta entre ela e cada um de seus vizinhos. Igualmente, há a probabilidade \( 1 - c \) de que a partícula retorne ao vértice \( q_i \). Nesta concepção, \( r(i, j) \) é igual por definição à \textit{probabilidade de transição estacionária} \( r_{i,j} \) de que a partícula se encontrará finalmente no vértice \( q_i \), ou:

\[ r(i, j) \triangleq r_{i,j} \]  

(2.22)

Formalmente, colocando-se todas as probabilidades \( r_{i,j} \) na matriz \( R = [r_{i,j}] \), então:

\[ R^T = cR^T \hat{G} + (1 - c)E \]  

(2.23)

onde \( E = [\vec{e}_i] \), \( 1 \leq i \leq Q \), é uma matriz \( N \)-por-\( Q \) de vetores unitários (apenas a linha \( i \) tem valor 1), \( c \) é a probabilidade de saída, e \( \hat{G} \) é a matriz de adjacência normalizada do grafo \( G \). A determinação de \( R^T \) pode ser feita por meio de um processo iterativo. Definidos os valores da matriz \( R^T \), prossegue-se combinando os valores de cada vértice \( j \) com relação aos vértices de interesse \( q_i \in \mathcal{Q} \), o que equivale à probabilidade de que todas as partículas que partiram dos vértices de interesse se encontrarão no vértice \( j \). O que é dado por:

\[ r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, \mathcal{Q}) = \prod_{i=1}^{Q} r(i, j) \]  

(2.24)

Uma vez calculada a adequação de cada vértice, torna-se possível calcular a adequação de cada subgrafo candidato, o que é feito por um algoritmo que segue à estratégia gulosa via programação dinâmica. Para isso, prossegue-se calculando a adequação de caminhos entre os vértices de interesse; o valor de adequação destes caminhos é armazenado (programação dinâmica), e consultado para que se
Capítulo 2. Compreensão de dados por meio de grafos

possia aumentar o tamanho destes caminhos ao mesmo tempo em que se aumenta o valor de adequação dos vértices selecionados (estratégia gulosa). Finaliza-se induzindo-se o subgrafo dado pela intersecção dos caminhos encontrados.

Figura 2.14: Sumarização de grafo integrada ao sistema GMine. (a) Grafo original; (b) destaque para um vértice em particular; (c) definição do conjunto de vértices de interesse; (d) subgrafo induzido sumarizando a conectividade relativa aos vértices de interesse.

A sumarização proposta foi integrada ao arcabouço de visualização GMine definindo uma sistemática de visualização para grafos, os quais poderiam ser analisados tanto de um ponto de vista macroscópico quanto com relação a detalhes dados por uma sumarização interativa, como detalhado no Apêndice F. O resultado vem de encontro ao fato de que uma rede complexa de alta magnitude não pode ser cognitivamente compreendida devido às limitações do ser humano (Cowan, 2010). De fato, há pouco sentido em se criar visualizações de grafos do tipo vértice-elo (node-link) quando a escala ultrapassa algumas centenas de vértices. Com a técnica proposta, pode-se reduzir a informação visual ao mesmo tempo em que se mantém o que é de interesse.

2.7 Visualização de grafos em larga escala

Grafos em larga escala se referem a grafos produzidos pela atividade humana globalizada, o que se tornou possível com as novas tecnologias de comunicação e computação. A visualização de tais grafos é um desafio, pois o grande volume de dados suplanta rapidamente a capacidade de exibição de dados dos dispositivos gráficos, e mesmo a capacidade de visualização dos seres humanos. A metodologia StructMatrix (Gualdron et al., 2015a) trata este problema por meio de duas técnicas. A primeira se baseia no princípio de que grafos são feitos de subestruturas que aparecem de maneira recorrente, incluindo aí cliques, núcleos bipartidos, estrelas, e cadeias. A segunda é a plotagem de uma matriz de adjacências com maior densidade de informações por pixel. Considerando este fato, trabalha-se com uma representação de grafo na qual linhas e colunas não se referem a vértices, mas sim a conjuntos de instâncias de estruturas, definindo um nível superior de abstração. A motivação é o fato de que analistas podem mais bem perceber significado a partir de um conjunto de estruturas simples extraídas da rede do

---

Esta técnica foi publicada no Fifth IEEE ICDM Workshop on Data Mining in Networks (Qualis A1, workshop), em 2015; sua concepção se baseou no trabalho de Mestrado do aluno Hugo Gualdron, orientado do autor deste documento.
que a partir de todos os vértices e arestas da rede, isto pois se obtém mais densidade de informação por unidade a ser analisada.

Nesta metodologia, considera-se um vocabulário com seis estruturas – veja a Figura 2.15, comumente consideradas em compreensão de grafos: estrelas falsas ($fs$), estrelas ($st$), cadeias ($ch$), quasi-10-cliques e cliques completos ($nc$, e $fc$), quasi-núcleos bipartidos e núcleos bipartidos completos ($nb$, e $fb$). Resumidamente o vocabulário de estruturas é dado pelo conjunto $\psi = \{fs, st, ch, nc, fc, nb, fb\}$.

![Figura 2.15: O vocabulário de estruturas considerado na metodologia StructMatrix; de (a) a (g), exemplos das estruturas consideradas.](image)

A metodologia inclui um algoritmo para detecção das subestruturas presentes em um dado grafo. O algoritmo se baseia no fato de que redes complexas possuem uma distribuição de vértices que obedece a uma lei de potência, com alguns poucos vértices com grau muito alto, e a maioria dos vértices com grau baixo; tais grafos também possuem uma componente conectada gigante. Intuitivamente, se forem considerados os vértices do grafo em ordem descendente de seus graus, se estará considerando uma ordem correspondente aos hubs mais importantes do grafo; se estes hubs forem progressivamente retirados, obtém-se uma eficiente técnica de varredura e particionamento do grafo (Kang e Faloutsos, 2011). Após cada retirada, surgem pequenas componentes conectadas (satélites), as quais podem ser classificadas segundo o conjunto $\psi$ ou aguardarem por uma nova iteração de retirada de hubs, até que todo o grafo seja processado e suas subestruturas sejam identificadas, definindo um conjunto de estruturas $S = \{s_0, s_2, \ldots, s_{|S|-1}\}$, $|S| = n$ classificadas de acordo com a função:

$$\text{type}(s) : S \rightarrow \psi$$

A partir daí plota-se uma matriz de adjacências $M_{n \times n}$, $0 < i < (n - 1)$ e $0 < j < (n - 1)$, entre estruturas; isto é, uma matriz que expressa as arestas definidas entre quaisquer duas estruturas do conjunto $S$. Formalmente, os elementos $m_{i,j}$ de $M_{n \times n}$ são definidos como:

$$m_{i,j} = \begin{cases} 1, & \text{se } D(s_i, s_j) > 0; \\ 0, & \text{do contrário.} \end{cases}$$

onde $D : S \times S \rightarrow \mathbb{N}$ é uma função que retorna o número de estruturas entre duas instâncias de estruturas.

Em seguida, a partir do conjunto $S$, a matriz terá um layout com múltiplas regiões, cada uma definida por dois tipos de estruturas ($\psi_i, \psi_j$), $\psi_i \in \psi$ e $\psi_j \in \psi$ – veja a Figura 2.16. Em cada região, mapeiam-se

---

10O conceito de quasi se refere a estruturas que possuem até 80% das arestas que uma estrutura completa possuiria.
os pontos da matriz de acordo com uma proporção direta, tendo-se o ponto \((x_{\min}, y_{\min})\) mapeado para o pixel na extrema esquerda inferior; e tendo-se o ponto \((x_{\max}, y_{\max})\) mapeado para o pixel na extrema direita superior; com os demais pontos de acordo com \((x, y) \rightarrow (\rho_x, \rho_y)\), como se segue:

\[
\rho_x = R(\psi_i, \psi_j) + \left(Res_x - 1\right) \frac{x - x_{\min}}{x_{\max} - x_{\min}} + \frac{1}{2}
\]

\[
\rho_y = R(\psi_i, \psi_j) + \left(Res_y - 1\right) \frac{y - y_{\min}}{y_{\max} - y_{\min}} + \frac{1}{2}
\]

onde \(R : \psi \times \psi \rightarrow \mathbb{N}\) é uma função que retorna o deslocamento em pixels da região \((\psi_i, \psi_j)\) e \(Res_x, Res_y\) define a resolução da projeção desejada.

**Figura 2.16:** Esquema da visualização StructMatrix.

Finalmente, plota-se também a cor de acordo com um mapa de calor para expressar o tamanho de duas estruturas conectadas. Neste ponto, os elementos \(m_{i,j}\) da matriz \(M_{n \times n}\) são dados por:

\[
m_{i,j} = \begin{cases} 
  C(NNodes(s_i) + NNodes(s_j)), & se D(s_i, s_j) > 0; \\
  0, & do contrário.
\end{cases}
\]

onde \(NNodes : S \rightarrow \mathbb{N}\) é uma função que retorna o número de vértices de uma dada estrutura, e \(C : \mathbb{N} \rightarrow [0.0, 1.0]\) é uma função que retorna um valor contínuo entre 0.0 e 1.0 de acordo com o número de vértices dados pela soma dos vértices de duas estruturas conectadas \(s_i, s_j \in S\).

Esta metodologia foi aplicada a grafos com mais de 5 milhões de arestas, os quais foram plotados em ordem de minutos. A técnica se mostrou mais escalável do que outras semelhantes, especialmente no que se fere à capacidade analítica proporcionada pelo uso de estruturas. De fato, houve duas contribuições, além da técnica de visualização, o algoritmo de detecção de estruturas trouxe avanços ao estado da arte. A metodologia peca pela necessidade de treinamento para que se interprete as visualizações; um viés ainda não resolvido para dados com a complexidade apresentada, mas que é comum em técnicas de visualização (Spence, 2001). O Apêndice G mostra mais detalhes incluindo o algoritmo proposto, exemplos de visualizações e análises, e testes de desempenho.
2.8 Visualização de Bancos de Dados Relacionais por meio de grafos

Bancos de Dados Relacionais são a solução de armazenamento de dados mais amplamente usada atualmente, como atesta o sítio de monitoramento de soluções de bancos de dados DB-Engines (http://db-engines.com/en/ranking), e como também atesta o autor Gary Anthes (Anthes, 2010). Tais bancos armazenam, por exemplo, dados institucionais a respeito de clientes, produtos, funcionários, e sobre todo tipo de controle que envolva operações transacionais. Muito embora tais bancos sejam adequados para armazenamento e recuperação de dados rigidamente estruturados, eles não são projetados para oferecer funcionalidades analíticas visuais.

No trabalho de Lima et al. (Lima et al., 2013), usou-se de grafos para representar os relacionamentos definidos entre as relações (tabelas) do Banco de Dados; o objetivo foi encontrar padrões, arranjos incomuns, e outras informações que possam auxiliar no suporte à decisão. Os grafos extraídos a partir das instâncias dos dados deram suporte a uma visualização interativa sobre um grafo hierárquico particionado, do tipo vértice-elo, na qual vértices e arestas representam, respectivamente, entidades e relacionamentos. A abordagem hierárquica particionada foi usada visando escalabilidade, gráfica e de processamento, pois a cardinalidade de bases de dados comumente é alta.

Na metodologia proposta, usou-se do conceito de SuperGrafo e de uma estrutura hierárquica capaz de representar um grafo particionado em disco, denominada Graph-Tree; esta estrutura foi necessária pois a organização hierárquica suporta a interação gráfica, enquanto que a representação de grafo modela a natureza dos dados. Formalmente:

Um SuperGrafo é recursivamente composto de vértices, SuperVértices (conjuntos de vértices), arestas, e SuperArestas (conjuntos de arestas), definido como:

Definição 2.2: [SuperGrafo] Dado um grafo finito e não direcionado \( G = \{V, E\} \), sem loops nem arestas paralelas, um Supergrafo é definido como: \( \overline{G} = \{\overline{V}, \overline{E}\} \), onde \( \overline{V} \) é um conjunto de SuperVértices \( \overline{v} \), e \( \overline{E} \) é um conjunto de SuperArestas \( \overline{e} \).

Definição 2.3: [SuperVértice] Um SuperVértice \( \overline{v} \) é recursivamente definido como um conjunto \( \overline{V}^\prime \) de SuperVértices ou vértices (se for uma folha), unido a um conjunto \( \overline{E}^\prime \) de SuperArestas \( \overline{e}^\prime \). Como se segue:

\[
\overline{v} = \{\overline{V}^\prime = \{\overline{v}_0, \overline{v}_1, ..., \overline{v}_{|\overline{V}^\prime|-1}\}, \\
\overline{E}^\prime = \{\overline{e}^\prime = \{e = (u, v) | u \in \overline{v}_i, v \in \overline{v}_j \subset \overline{V}^\prime\}\}\}
\]  
\(2.29\)

Definição 2.4: [SuperAresta] Uma SuperAresta \( \overline{e}^\prime \) representa todas as arestas \((u, v) \in E\) que conectam vértices de um SuperVértice \( \overline{v}_i \) a vértices de um SuperVértice \( \overline{v}_j \). Formalmente, a SuperAresta entre os SuperVértices \( \overline{v}_i \) e \( \overline{v}_j \) é definida da seguinte maneira:

\[
\text{SuperAresta}(\overline{v}_i, \overline{v}_j) = \overline{e}^\prime = \{e = (u, v) | (u, v) \in E, u \in \overline{v}_i, v \in \overline{v}_j\}
\]  
\(2.30\)


 A estrutura Graph-Tree foi proposta durante o doutoramento como uma representação de dados em memória; neste trabalho, ela foi aperfeiçoada como uma estrutura para acesso a disco, adequada a soluções para Bancos de Dados.
Capítulo 2. Compreensão de dados por meio de grafos

Definição 2.5: [VérticeEmAberto] Um vértice \( v \in V \) é denominado VérticeEmAberto se \( \exists e = (v, u) \), tal que \( u \notin v \).

![Figura 2.17: SuperGrafo obtido a partir de um grafo particionado.]

A Figura 2.17 exemplifica a abstração SuperGrafo. A Figura 2.17(a) mostra um grafo \( G \), definido como \( G = \{V = \{1, 2, 3, 4, 5, 6, 7, 8\}, E = \{(1, 7), (1, 8), (2, 7), (3, 4), (3, 7), (4, 5), (4, 6), (7, 8)\}\} \).

A partir do grafo \( G \) é possível conceber o particionamento hierárquico apresentado como SuperGrafo \( \mathcal{G} \) na Figura 2.17(b). Este particionamento é composto de SuperVértices \( \pi_0 \) a \( \pi_6 \) e das SuperArestas correspondentes:

- \( \pi_0 = \{V = \{v_0, v_1\}, E = \{\pi_{12} = \{(3, 7)\}\}\} \)
- \( \pi_1 = \{V = \{v_1, v_2\}, E = \{\pi_{14} = \{(1, 7), (1, 8), (2, 7)\}\}\} \)
- \( \pi_2 = \{V = \{v_3, v_4\}, E = \{\pi_{35} = \{(4, 5), (4, 6)\}\}\} \)
- \( \pi_3 = \{V = \{v_0, v_1\}, E = \{\}\} \)
- \( \pi_4 = \{V = \{v_7, v_0\}, E = \{\pi_{14} = \{(7, 8)\}\}\} \)
- \( \pi_5 = \{V = \{3, 4\}, E = \{\pi_{35} = \{(3, 4)\}\}\} \)
- \( \pi_6 = \{V = \{5, 6\}, E = \{\}\} \) \((2.31)\)

A Figura 2.17(c), por sua vez, apresenta a correspondente estrutura de árvore, a qual reflete o particionamento hierárquico do SuperGrafo. Na figura, pode-se ver que a estrutura é projetada de maneira que os SuperVértices folha são seletivamente carregados a partir do disco. A principal característica da estrutura é sua habilidade em determinar dinamicamente as arestas que interconectam vértices, ou SuperVértices. Esta característica implica que:

1. dado um vértice (ou tupla), pode-se determinar todas as arestas que se conectam a este vértice sem ter que checar todas as partições em todos os níveis da hierarquia – para isto basta checar os vértices em aberto dos SuperVértices ao longo do caminho entre um dado vértice e a raiz da hierarquia, confrontando-os (por meio de intersecção) com os vértices em aberto de seus SuperVértices irmãos;
2. dado um SuperVértice (ou conjunto de tuplas), pode-se determinar todas as arestas (SuperAresta) que se conectam a este conjunto.

Estas duas características são essenciais para a metodologia de exploração visual interativa, pois elas permitem a inspeção da informação relacional da base de dados.

Após a formalização de SuperGrafo, o passo seguinte da metodologia é definir um método para se criar um grafo hierarquicamente particionado a partir de uma Base de Dados Relacional, o que leva a uma
2.8. Visualização de Bancos de Dados Relacionais por meio de grafos

representação de SuperGrafo de maneira visual interativa. Dada uma base de dados, o correspondente grafo hierarquicamente particionado é definido de acordo com as seguintes diretrizes:

- os vértices do grafo correspondem a instâncias de entidades (tuplas) da base;
- as arestas correspondem a chaves estrangeiras entre instâncias de entidades;
- o número de níveis da hierarquia é dado pelo número de atributos de interesse, a ser definido pelo usuário;
- o número de partições é dado em função do tipo do atributo, categórico, nominal, ou numérico; por exemplo, pela análise da distribuição dos dados de cada atributo, dividindo-se os dados de acordo com percentis estatísticos – ilustrado na Figura 2.18; ou pelo significado semântico dos valores, como localização (país ou cidade).

A Figura 2.18: Particionamento dado por percentis estatísticos; neste caso, múltiplos de 5 para o atributo Idade de uma entidade Pessoa.

A Figura 2.19 mostra um exemplo simples deste processo. Na figura, as entidades Pessoa e Publicação definem vértices, o relacionamento Autoria define arestas, os atributos Idade, e Cidade, definem os níveis de Pessoa, e os atributos País, e Ano definem os níveis de Publicação. As partições de Idade são dadas por uma semântica específica, 50 anos; as partições de Ano, por um ano específico, o de 2006; ao passo que Cidade e País, são particionados de acordo com a representatividade geográfica de cada valor.

A partir do grafo particionado e representado como um SuperGrafo, define-se a visualização hierárquica interativa. A Figura 2.20 mostra a visualização para o exemplo da Figura 2.19. A visualização inicia com um SuperVértice para cada entidade representada como um círculo, Figura 2.20(a). Neste nível é possível verificar que há 13 relacionamentos entre Pessoa e Publicação. No próximo nível, verificam-se os particionamentos dados por idade e ano, Figura 2.20(b); pode-se ler que as pessoas com mais de 50 anos não publicaram após o ano de 2006. Essa mesma lógica prossegue nos demais níveis e particionamentos, sendo que as propriedades da estrutura, e do ambiente visual, permitem combinar quaisquer SuperVértices, de quaisquer níveis. A metodologia também permite que os detalhes das arestas e dos vértices sejam verificados sob demanda.

A metodologia é particularmente adequada à compreensão básica de um esquema Relacional sem a necessidade de escrita de SQL, e de maneira visual. Demonstra-se que é possível, mesmo em bases de dados em alta escala, perceber a distribuição das relações, dos relacionamentos, e dos domínios de atributos, combinando-se as informações entre relações, ou tuplas, via cálculos de conectividade. Análises
Capítulo 2. Compreensão de dados por meio de grafos

Figura 2.19: SuperGrafo construído a partir do particionamento dado por duas entidades relacionais ligadas por um relacionamento muitos-para-muitos.

envolvendo esta metodologia, incluindo um estudo de caso sobre a base de dados da Universidade de São Paulo (Sistema Tycho-USP, com 1.503.389 de tuplas e 1.868.443 de relacionamentos), e testes de desempenho para computação das arestas requisitadas pelo usuário – em comparação com operações de agregação em um Banco de Dados Relacional, são apresentados no Apêndice H. No entanto, a metodologia apresentada foca nos relacionamentos dos dados, não oferecendo muitas funcionalidades no que se refere à compreensão das informações de uma dada instância de relação (uma única tabela). Na próxima seção, será apresentada uma metodologia que trabalha sobre esta vertente de análise visual de dados.

2.9 Hierarchical visual filtering

Com o intuito de promover a inspeção de relações (não de relacionamentos) em bases de dados, o trabalho de Rodrigues-Jr et al. (RODRIGUES-JR et al., 2013a) 13 – Apêndice I – descreve o uso de visualizações heterogêneas em um ambiente com múltiplas visões fazendo uso de ações pragmáticas e epistêmicas em uma técnica denominada Hierarchical Visual Filtering. No trabalho, ações pragmáticas correspondem à filtragem iterativa hierárquica, ao passo que ações epistêmicas correspondem às possibilidades de registrar, anotar, e recuperar visualizações intermediárias criadas no processo pragmático.

Formalmente, como ilustrado na Figura 2.21, o processo é composto por uma relação $D$, uma visualização $V$, uma função de visualização $v$, uma função interativa $f$, e uma função $\Lambda$ que opera o inverso da função $v$.

Na Figura 2.21(b), a função de visualização $v : D_i \rightarrow V_i$ é parametrizada por um par $(s, g)$ onde $s$ é o esquema de espacialização que determina como os dados da relação $D$ ocupam o espaço da visualização (projeção, sequência, correspondência, entre outras possibilidades), e $g$ é o conjunto de marcas gráficas (pontos, linhas, curvas, ícones, entre outras possibilidades) sobre $V_i$. Já a função de filtragem interativa

2.9. Hierarchical visual filtering

Figura 2.20: Exemplo de uma visualização hierárquica sobre uma Base de Dados Relacional.

\[ f : V_i \rightarrow V_i' \] é parametrizada por um par \((d, e)\) onde \(d\) é um conjunto de dimensões dos dados, e \(e\) é um conjunto de predicados relacionais que se aplicam a \(d\) para determinar a seleção de dados. A filtragem interativa produz uma configuração alterada \(V_i'\) composta de um subconjunto de entidades gráficas de \(V_i\). Por último, a função \(\Lambda : V_i' \rightarrow D_{i+1}\) recebe \(V_i'\) e opera o inverso da função \(v\) usando os mesmos parâmetros, e retornando a relação \(D_{i+1}\), \((D_{i+1} \subseteq D_i)\), a qual definirá a visualização \(V_i'\).

A Figura 2.21(c) apresenta o ciclo iterativo que define a técnica Hierarchical Visual Filtering. De acordo com este ciclo, uma relação \(D_i\) é usada para criar uma visualização \(V_i\) por meio da função \(v\). Esta visualização é interativamente filtrada por meio da função \(f\) para determinar a nova configuração \(V_i'\). Após isso, os dados sendo apresentados em \(V_i'\) são extraídos por meio da função \(\Lambda\). Com os dados extraídos de \(D_{i+1}\), uma nova visualização \(V_{i+1}\) é criada. Este processo repete-se iterativamente de acordo com os objetivos exploratórios do usuário, caracterizando as ações pragmáticas do método. As ações epistêmicas se referem ao registro dos passos intermediários \((V_0, V_1, ..., V_n)\), e dos respectivos parâmetros das funções \(v\) e \(f\).

A Figura 2.22 ilustra a aplicação do método sobre uma relação com dados advindos da empresa Embrapa (Empresa Brasileira de Pesquisa Agropecuária). O método permite que uma mesma relação seja filtrada inúmeras vezes, Figura 2.22(b), definindo uma árvore de sub visualizações, cada qual com um conjunto de dados próprio. Cada visualização, por sua vez, pode passar por um novo ciclo de filtragem – Figura 2.22(c), o que leva à criação de uma hierarquia. Então, cada visualização torna-se nó de uma árvore (grafo acíclico) e arestas são desenhadas para explicitar os relacionamentos entre conjuntos e subconjuntos filtrados. Cada visualização pode ser objeto de análise interativa, e receber anotações dos usuários; além disso, os detalhes que definem cada ciclo são registrados e apresentados sob demanda.

Mais detalhes sobre a análise dos dados da Embrapa, validações com usuário, prototipação, e gerenciamento do espaço de visualização são apresentados no Apêndice I. As contribuições do trabalho...
Figura 2.21: Hierarchical Visual Filtering. (a) Elementos, (b) funções constituintes, e (c) ciclo iterativo.

incluem: (1) a redução da sobreposição de elementos gráficos, aumentando a escala; e (2) a redução da sobrecarga cognitiva, especialmente da memória.

2.10 Considerações finais

Neste capítulo foram delineadas contribuições a respeito da área mais ativa de trabalho do candidato. Nem todas as contribuições descritas são necessariamente as mais importantes de sua carreira mas, em conjunto, definem um tema único adequado para descrever as atividades do candidato após o Doutorado. No capítulo que se segue, em complementação, serão descritos trabalhos da segunda área de atuação do candidato. Apesar de ser um capítulo de complementação, as contribuições descritas são de igual, ou de maior, relevância que as demais; apenas não foram mais enfatizadas por uma opção de organização e conteúdo considerada mais adequada ao tempo e espaço disponíveis.
2.10. Considerações finais

Figura 2.22: Definição de uma análise hierárquica sobre dados da Embrapa. (a) Visualização $V_0$ de todos os dados da relação $D_0$. (b) Filtragem interativa gerando 5 subconjuntos $D_i \subseteq D_0, 1 \leq i \leq 5$, e correspondentes visualizações $V_i, 1 \leq i \leq 5$: cada uma com dois conjuntos de parâmetros $s, g$ definindo as visualizações coordenadas paralelas e projeção multidimensional. (c) Novo ciclo iterativo definindo as visualizações $V_i, 6 \leq i \leq 9$. 

Embrapa

| Precip | MaxT | MinT | NDVI | WRSI | ET | ET | ET | ET |

(a)

(b)

(c)
Capítulo 2. Compreensão de dados por meio de grafos
3.1 Considerações iniciais

Além do processamento analítico de dados representáveis como grafos, linha mestra deste texto, o candidato também trabalha com Gerenciamento de Dados baseado em Conteúdo, uma outra vertente da grande área de Bancos de Dados. Essa tendência decorre de sua formação (Mestrado e Doutorado) junto ao Grupo de Bases de Dados e Imagens (GBDI) do ICMC-USP, cujo principal tema de trabalho é o Gerenciamento de Dados baseado em Conteúdo.

3.2 Conceitos sobre Gerenciamento de Dados baseado em Conteúdo

Além de números e caracteres, Bancos de Dados são comumente usados para gerenciar informações de maior complexidade, como imagens, vídeos, e textos não estruturados. Estes dados não podem ser manipulados (indexados e recuperados) diretamente pois, neste caso, a comparação direta entre valores não faz sentido, além de ser ineficiente. Para tratar este problema, o procedimento mais aceito na literatura é converter os dados complexos de maneira que possam ser expressos como vetores numéricos; neste formato eles podem ser comparados por meio de funções. A conversão dos dados em vetores é baseada nas características do conteúdo dos dados, sendo que estes vetores determinam os assim chamados vetores de características. Estes conceitos, definidos a seguir, são usados nas próximas seções deste capítulo.

Formalmente, dado um elemento $d$ pertencente a um domínio de dados complexos $D$ – por exemplo imagens, o primeiro passo para se manipular este dado é usar uma função $f : D \rightarrow \mathcal{D}$, onde $\mathcal{D} \subseteq \mathbb{R}^n$
é um espaço de características n-dimensional. O resultado desta conversão, denominada *extração de características*, é um vetor $\mathbf{v} = \{x_0, x_1, ..., x_{n-1}\}$ com n números representativos intrínsecos ao dado original.

Desta maneira, dado um conjunto de dados complexos $D' \subseteq D$ e seu correspondente conjunto de vetores extraídos $D' \subseteq \mathcal{D}$, é necessário um método para se calcular a distância (dissimilaridade) entre quaisquer dois vetores. Para isso, usa-se uma função do tipo $\delta: \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}^+$ que retorna um número real correspondendo à distância entre os vetores. Tendo-se a distância entre qualquer par de elementos, torna-se possível determinar uma relação de ordenem entre um dado elemento do conjunto e os demais. A distância Minkowski ponderada (Wilson e Martinez, 1997) é uma das mais usadas:

$$\delta_{Minkowski}(\mathbf{x}, \mathbf{y}) = \sqrt[p]{\sum_{i=0}^{n-1} w_i (x_i - y_i)^p} \quad (3.1)$$

onde $\mathbf{x} = \{x_0, x_1, ..., x_{n-1}\}$ e $\mathbf{y} = \{y_0, y_1, ..., y_{n-1}\}$ são vetores com n características numéricas, $o_i$ e $o_j \in \mathcal{D}$; $p$ é a ordem da distância (geralmente $p \in \{0, 1, 2\}$); e o vetor $\mathbf{w} = \{w_0, w_1, ..., w_{n-1}\}$ é um vetor com n pesos, de maneira a ponderar a importância de cada uma das dimensões de características.

Com um conjunto de vetores de características e uma função distância, define-se um espaço métrico; formalmente definido como um par $M = < \mathcal{D}, \delta() >$ sendo que, dados três elementos quaisquer $o_i, o_j$ e $o_k \in \mathcal{D}$, os seguintes axiomas devem ser satisfeitos:

1. **Simetria**: $\delta(o_i, o_j) = \delta(o_j, o_i)$

2. **Não negatividade**: $0 \leq \delta(o_i, o_j) < \infty$ para $o_i \neq o_j$; e $\delta(o_i, o_i) = 0$

3. **Desigualdade triangular**: $\delta(o_i, o_j) \leq \delta(o_i, o_k) + \delta(o_k, o_j)$

A potencial analogia espacial é o que torna o conceito de espaço métrico tão útil para os seres humanos, os quais podem trabalhar por meio da noção intuitiva de similaridade; usualmente usada da seguinte maneira:

**Definição 3.1:[Consulta por abrangência]**: dado um objeto $o_q$, e uma abrangência (raio) $r_q$, a consulta por abrangência corresponde ao conjunto dado por $RangeQuery(o_q, r_q) = \{o_n | o_n \in \mathcal{D}, \delta(o_n, o_q) \leq r_q\}$

**Definição 3.2:[Vizinho mais próximo]**: dado um objeto $o_q$, o vizinho mais próximo de $o_q$ corresponde ao elemento $NNQuery(o_q) = \{o_n | o_n \in \mathcal{D}, \delta(o_q, o_n) \leq \delta(o_q, o_i) \forall o_i \in \{\mathcal{D} - o_n\}\}$.

A definição 3.2 extrapola de maneira trivial ao conceito de k vizinhos mais próximos, $k > 1$, produzindo uma lista ordenada na qual o $(n-1)$-ésimo elemento é mais próximo a $o_q$, ou está à mesma distância, que o n-ésimo elemento.

A maneira simples de se computar tais consultas exige o cálculo da distância entre um dado elemento e todos os demais. Uma solução mais elaborada é o uso de *métodos de acesso métrico*, estruturas de indexação que só precisam conhecer a distância entre os pares de elementos e que são especialmente projetadas para resolver consultas por similaridade em tempo logarítmico. Estas estruturas se baseiam em elementos, ditos representantes, do espaço de dados a partir dos quais um raio de cobertura é calculado; estes elementos têm como elementos filhos aqueles dentro de seu raio. Este mecanismo é usado na forma de uma árvore – ilustrada na Figura 3.1 – capaz de, progressivamente, reduzir o espaço de busca de
3.3 Visualização de espaços de características

Espaços métricos advindos de conjuntos de dados complexos possuem um apelo natural à visualização; mais especificamente, às projecções multidimensionais de dados, as quais, similarmente, se constroem por meio de distâncias entre elementos. De fato, projecções multidimensionais são representações gráficas da similaridade entre elementos de informação. No trabalho *Combining Visual Analytics and Content Based Data Retrieval Technology for Efficient Data Analysis* (RODRIGUES-JR et al., 2010)\(^1\) descreve-se um arcabouço denominado *Metric Space Platform* (http://www.icmc.usp.br/pessoas/junio/MetricSPlat/index.htm) voltado à definição de espaços métricos, à recuperação de dados por conteúdo, e à visualização dinâmica de tais funcionalidades.

Considerando estes conceitos, o arcabouço *Metric Space Platform* torna disponível uma tecnologia na qual é possível definir qualquer função de distância combinada com qualquer método de acesso métrico – foram usados a *Slim-tree* (Traina et al., 2000), e a *DBM-tree* (Vieira e Traina Jr., 2004), sem a necessidade de recompilação do sistema; o que é realizado por meio de uma *Application Programming Interface* acessada via *Dynamic Linking Library (DLL)*. Estas funcionalidades são combinadas com cinco técnicas de visualização: Parallel Coordinates (Inselberg e Dimsdale, 1990), Scatter Plots, Table Lens (Rao e Card, 1994), Star Coordinates (Kandogan, 2000), e projeção multidimensional via FastMap (Faloutsos e Lin, 1995). Neste trabalho, a técnica FastMap é a mais importante, pois possibilita a projeção dos dados via redução de dimensionalidade de \(n\) para 3 dimensões, em um espaço interativo 3D. Esta técnica tem como objetivo encontrar uma redução de dimensionalidade \(m : \mathbb{D} \rightarrow \mathbb{R}^3\) que minimize a soma de diferenças entre a distância \(\delta\) no espaço original \(n\)-dimensional e a distância \(d\) no espaço de projeção 3-dimensional, isto é:

\[
\text{Argmin}_m \left( \sum_{i,j} |\delta(o_i, o_j) - d(m(o_i), m(o_j))| \right) \tag{3.2}
\]

onde \(o_i\) e \(o_j\) \(\in\) \(\mathbb{D}\).

O resultado é a visualização dos dados como pontos em um espaço no qual se percebe visualmente a distância que os diferencia. O sistema permite inspecionar um Banco de Dados qualquer, criar espaços

\(^1\)Este trabalho foi publicado nos *Proceedings of the International Conference on Information Visualisation* (Qualis B1), em 2010.

---

Figura 3.1: *Slim-tree* indexando 17 objetos.
Figura 3.2: Visualização do resultado de uma consulta por similaridade em um ambiente 3D. (a) Centro da consulta posicionado na origem do espaço Euclidiano de projeção. (b) Interação 3D. (c) Funcionalidade de zoom.

O sistema Metric Space Platform foi usado para a compreensão de dados advindos da empresa Embrapa passando por experimentos que focaram em avaliação, via estudo de caso, da parametrização da função distância Minkowski usada na projeção e nas consultas. Desta maneira, o analista foi capaz de definir dinamicamente, baseando-se em feedback visual, quais dimensões eram de maior importância. Os resultados demonstraram potencial no que se refere à descoberta de fatores não óbvios; foi possível, por exemplo, identificar regiões de produção de cana de açúcar considerando-se fatores como chuva e evaporação do solo, além de se identificar agrupamentos de acordo com fatores de índices de vegetação adquiridos via satélite. Além desta contribuição, este trabalho contribuiu ao descrever a integração de um amplo conjunto de técnicas em uma plataforma disponibilizada à comunidade em http://www.icmc.usp.br/pessoas/junio/MetricSPlat/index.htm.

3.4 Otimização do espaço de características

Um dos problemas relacionados ao uso de espaços métricos para a recuperação de dados por meio de consultas por similaridade é o fato de que as funções distância nem sempre correspondem à compreensão, e correspondente expectativa de similaridade, do usuário. Este problema é comumente denominado “lacuna semântica” (Wang et al., 2008). No trabalho intitulado Image Retrieval Employing Genetic Dissimilarity Weighting and Feature Space Transformation Functions (Avalhais et al., 2012)², trata-se a lacuna semântica por meio de duas técnicas: algoritmos genéticos e realimentação de relevância. No trabalho, provida uma família de funções de ponderação das características – Figura 3.3, usa-se de relevance feedback para se guiar a convergência de algoritmos genéticos, os quais devem encontrar a melhor combinação de funções de ponderação. A metodologia foi aplicada sobre conjuntos de imagens, apresentando resultados promissores de acurácia na recuperação de imagens por similaridade. A grande contribuição do trabalho, no entanto, foi o uso de funções de ponderação – ao invés de constantes de ponderação. O uso de funções torna o espaço de características mais versátil, isto pois alguns valores do domínio de uma determinada característica podem ser de maior relevância (maior ponderação, portanto) do que outros valores do mesmo domínio. Essa versatilidade torna-se ainda maior quando se usam combinações de funções; no entanto, tem-se também maior complexidade. Nesse caso, torna-se necessário

²Este trabalho foi publicado no periódico ACM SIGAPP Applied Computing Review, em 2012; sua concepção se baseou no trabalho de Mestrado da aluna Letrícia Avalhais, co-orientada do autor deste documento.
3.5 Detecção de fogo em imagens

um método de aprendizado de máquina para a escolha das funções. No trabalho proposto, optou-se por algoritmos genéticos.

![Figura 3.3: Conjunto de funções usadas na metodologia proposta (Avalhais et al., 2012).](image)

3.5 Detecção de fogo em imagens

Um dos objetivos do processamento baseado em conteúdo é interpretar, com alguma semântica, o que é expresso nos dados. Este processo ainda não é possível de maneira generalizada, para qualquer tipo de dado, e em qualquer contexto; mas, para situações específicas, torna-se possível ter o computador “entendendo” o que é expresso em um dado de maior complexidade, como uma imagem. Esta possibilidade é investigada no trabalho *Techniques for effective and efficient fire detection from social media images* (Bedo et al., 2015)\(^3\). Este trabalho foi realizado no contexto do projeto Rescuer, em parceria com a União Europeia, o qual objetiva o desenvolvimento de técnicas computacionais aplicáveis a situações de emergência – mais detalhes em [http://www.rescuer-project.org](http://www.rescuer-project.org). O candidato é colaborador deste projeto atuando como coordenador executivo da equipe do GBDI.

A metodologia proposta introduz uma arquitetura – Figura 3.4, denominada *Fast-Fire Detection* (FFireDt), a qual integra técnicas de extração de características de imagens, funções distância, suporte de um sistema gerenciador de Bancos de Dados, e classificação via aprendizado de máquina. Esta arquitetura é capaz de processar, com alto desempenho, imagens advindas de um fluxo constante de dados, um dos requisitos do projeto Rescuer. A maior contribuição do trabalho, além do projeto e da instanciamento da arquitetura, foi a sistemática experimentação envolvendo 36 combinações de extratores de características com funções de distância sob 5 medidas de desempenho. No universo de processamento baseado em conteúdo, trata-se de uma contribuição relevante, isto pois não é trivial determinar qual é a melhor maneira de se preencher a lacuna semântica para um determinado domínio. Desta maneira, o trabalho proveu elucidação sobre o domínio de imagens contendo fogo.

\(^3\)Este trabalho foi publicado nos *Proceedings of the International Conference on Enterprise Information Systems* (Qualis B1), em 2015; sua concepção se baseou no trabalho de Doutorado do aluno Marcos Bedo e de outros colaboradores.
Figura 3.4: Arquitetura FFireDt. O módulo de avaliação recebe uma imagem não rotulada, cria sua representação na forma de um vetor de características, e a classifica usando classificação Instance-Based Learning. O sistema classifica a imagem e provê ao especialista a possibilidade de recuperar imagens semelhantes.

Em continuação ao objetivo de se detectar fogo com acurácia, foi também desenvolvido o trabalho BoWFire: Detection of Fire in Still Images by Integrating Pixel Color and Texture Analysis (Chino et al., 2015). Neste segundo desenvolvimento, o objetivo era reduzir o número de falsos positivos apresentados no melhor resultado do sistema FFireDt. A ideia foi combinar extratores de características de diferentes naturezas. Foram usados extratores baseados em cor e em textura (Florindo et al., 2015) ao mesmo tempo, de maneira que uma imagem seria classificada como contendo fogo apenas se ambas as classificações assim o indicassem. Os resultados representados via curvas ROC indicaram significativo ganho de precisão no método, o qual foi integrado ao sistema Rescuer que deve entrar em operação em 2016.

Figura 3.5: Exemplo da aplicação da técnica de Superpixel sobre uma imagem contendo fumaça.

4Este trabalho foi publicado nos Proceedings of the 28th SIBGRAPI Conference on Graphics, Patterns and Images (Qualis B1), em 2015; sua concepção se baseou no trabalho de Doutorado do aluno Daniel Chino e de outros colaboradores.
3.6 Detecção de fumaça em imagens

No mesmo contexto do projeto Rescuer que motivou o trabalho apresentado na Seção 3.5, o trabalho *Unveiling smoke in social images with the SmokeBlock approach* (Cazzolato et al., 2016) objetiva a detecção de imagens contendo traços de fumaça em imagens produzidas por meio de crowdworking. A metodologia denominada SmokeBlock determina um processo completo de pré-processamento, extração de características, e classificação de imagens com relação à presença de fumaça. A maior inovação da proposta é o uso da técnica de Superpixels (Achanta et al., 2012), a qual reduz a redundância e a complexidade da informação contida em uma imagem ao agrupar pixels adjacentes que possuam similaridade de cor. Ao se agrupar pixels semelhantes – veja a Figura 3.5, torna-se possível processar um número menor de unidades de informação, o que reduz o custo de processamento e, em alguns casos, melhora a precisão das técnicas baseadas em conteúdo. Os resultados da técnica SmokeBlock são promissores, especialmente se comparados com outras propostas encontradas na literatura. Todavia, a acurácia da classificação foi notadamente menor do que a da metodologia FFireDt, pois a detecção de fumaça é um desafio mais complexo. Um dos próximos desenvolvimentos é usar os princípios da técnica BoWFire (Seção 3.5) para se obter mais precisão na detecção de fumaça ao se aplicar múltiplos extratores simultaneamente.

3.7 Projeto e desenvolvimento de um sistema de Gerenciamento de Dados baseado em Conteúdo

Todas as técnicas do projeto Rescuer, incluindo as que foram discutidas nas Seções 3.5 e 3.6, foram instanciadas de acordo com uma arquitetura inovadora denominada *Data-Centric Crisis Management* (DCCM). A arquitetura DCCM define a integração entre técnicas de Gerenciamento de Dados baseado em Conteúdo e um Banco de Dados Relacional com ênfase em situações de emergência, contexto do projeto Rescuer. Dentre as inovações propostas estão uma extensão da linguagem SQL, consultas por similaridade, classificação, filtragem de dados semelhantes e, dado um conjunto de imagens, geração de um histórico de imagens representativas. A DCCM foi desenvolvida como uma extensão do Banco de Dados de código livre PostgreSQL (http://www.postgresql.org/), resultando em uma versão deste software com funcionalidades de classificação e recuperação de dados baseada em conteúdo. Este esforço é descrito no trabalho intitulado *On the Support of a Similarity-Enabled Relational Database Management System in Civilian Crisis Situations* (Oliveira et al., Submetido em Dez/2015), no qual se reportam experimentos de acurácia e de eficiência (desempenho) para situações esperadas em emergências.

3.8 Considerações finais

Esta seção apresentou resultados da área de Gerenciamento de Dados baseado em Conteúdo, na qual o candidato é bastante ativo. O objetivo foi demonstrar seu perfil de trabalho de maneira complementar, o qual também se concentra em técnicas de gerenciamento de dados, além de técnicas de análise de dados. Os resultados apresentados advém, sobretudo, do projeto Rescuer, no qual o candidato tem

---


6 Este trabalho foi submetido ao evento *International Conference on Enterprise Information Systems* (Qualis B1), em Dez/2015; sua concepção se baseou nos trabalhos de Doutorado dos alunos Paulo Oliveira e Hugo Gualdron (orientado do candidato), além de outros colaboradores.
Capítulo 3. Gerenciamento de Dados baseado em Conteúdo

atuado desde 2014. Este projeto tem sido o mais desafiador até o momento deste trabalho de Livre Docência; isto pois ele abrange equipes de outras instituições no Brasil (Universidade Federal da Bahia) e no exterior (Universidade Politécnica de Madrid, Espanha; Instituto Fraunhofer, Alemã; e German Research Center for Artificial Intelligence, Alemã), e propõe metas cobradas sistematicamente pelo comitê avaliador da Comissão Europeia de acordo com o fundo de inovação e pesquisa FP7 (https://ec.europa.eu/research/fp7/index_en.cfm).
Conclusões

Este documento apresentou um texto sistematizado descrevendo a parte da obra do candidato que se refere, principalmente, à análise de bases de dados representáveis como grafos, mas que, também, se refere ao tema de Gerenciamento de Dados baseado em Conteúdo. A descrição dos trabalhos, disponíveis no apêndice, foi organizada em seções cuja ordem se deu de acordo com temas semelhantes no que se refere às contribuições do candidato: algoritmos para processamento de grafos em larga escala, processamento analítico de grafos para detecção de padrões, visualização tendo-se o modelo de grafos como base, e Gerenciamento de Dados baseado em Conteúdo. A descrição dos trabalhos foi precedida de motivação circunstanciada, seguindo-se um breve resumo dos resultados; o intuito foi apresentar os trabalhos descritos no apêndice de maneira sucinta, mas contextualizada e perfazendo uma única obra. Nesta seção de conclusões, adicionalmente, cabe um resumo das contribuições nas linhas de pesquisa apresentadas:

Desenvolvimento algorítmico

- proposição de algoritmos de altíssima eficiência para processamento sobre um único nó de processamento;
- projeto e desenvolvimento de um arcabouço de processamento sobre um único nó capaz de suplantar o desempenho de todos os arcabouços semelhantes propostos na literatura.

Processamento analítico

- proposição de uma metodologia algébrica para o processamento matricial de grafos segundo um processo de mineração de dados;
Capítulo 4. Conclusões

- demonstração do uso de múltiplas técnicas de análise, baseadas em métricas extraídas de grafos, com o fim de se caracterizar e compreender redes de co-autoria;
- proposição de uma técnica baseada em processos Markovianos para a sumarização de grafos guiada pelo interesse do usuário.

Visualização

- projeto de uma técnica de visualização altamente escalável baseada em matrizes de adjacência densas produzidas a partir de um algoritmo de detecção de subestruturas recorrentes;
- projeto de uma técnica de visualização hierárquica de grafos extraídos de Bancos de Dados Relacionais considerando múltiplos relacionamentos e entidades;
- proposição de uma técnica de interação iterativa capaz de particionar uma relação de uma base de dados com o objetivo de se ampliar a capacidade analítica, relacionando subconjuntos de dados entre si.

Gerenciamento de Dados baseado em Conteúdo

- projeto de uma metodologia capaz de integrar técnicas de Recuperação de Dados baseada em Conteúdo e técnicas de Visualização de Informações;
- desenvolvimento de técnicas para aprimorar a Recuperação de Dados baseada em Conteúdo envolvendo feedback do usuário processado via algoritmos genéticos;
- proposição de técnicas e sistemas para detecção, via classificação, de fogo e de fumaça em imagens objetivando a aplicação sobre um problema real de suporte a situações de crise.

A produção aqui reportada teve colaboração, sobrecarregada, de parceiros do próprio ICMC-USP, da Universidade Georgia Tech, GA, EUA (onde um dos alunos de Mestrado fez estágio Fapesp-BEPE), e da Universidade Carnegie Mellon, PA, EUA (a qual mantém colaboração de pesquisa com o GBDI, e onde o candidato fez uma visita técnica de um mês em 2014). Ressalta-se que a produção do candidato tem crescido ano a ano, ao passo que sua experiência aumenta, e que mais alunos trabalham sob sua orientação. Com efeito, a partir de janeiro de 2016, o candidato se tornou orientador pleno em nível de Doutorado e Mestrado junto ao Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional do ICMC-USP.

Desenvolvimentos futuros e em andamento

Atualmente o candidato orienta três alunos de Doutorado e três de Mestrado. Dois dos alunos de Doutorado, e um de Mestrado trabalham com os temas apresentados neste trabalho. Em síntese, um dos trabalhos de Doutorado dá continuidade ao desenvolvimento de algoritmos paralelos assíncronos. Mais especificamente, está se desenvolvendo um algoritmo para a computação das probabilidades dadas pela técnica de belief propagation. A ideia é usar de propriedades algébricas para calcular de maneira aproximada o estado de convergência do algoritmo, mas ainda sim com resultados próximos do exato. O outro trabalho de Doutorado está explorando novas possibilidades do arcabouço M-Flash, como o
deslocamento da hierarquia de memória de modo a usar apenas o cache e a memória principal, permitindo desempenho superior até mesmo às melhores bibliotecas de gráficos que funcionam sobre matrizes ou listas de adjacência. O aluno de Mestrado está trabalhando na linha de extração de métricas, locais e globais, que caracterizem gráficos; estão sendo usados gráficos de redes de locomoção viária extraídos do serviço OpenStreetMaps (https://www.openstreetmap.org) de diferentes continentes. Dentre as medidas usadas estão o PageRank, caminhos mínimos, e diâmetro. O objetivo é levantar hipóteses a respeito de tendências de planejamento viário, possivelmente prevendo problemas e melhores práticas.

Os outros trabalhos de Doutorado e de Mestrado trabalham com técnicas de Gerenciamento de Dados baseado em Conteúdo. O aluno de Doutorado trabalha com extração de características de folhas de soja com o intuito de identificar tipos de doenças, e detectar computacionalmente ataques de insetos. Resultados promissores, com publicação em segunda fase de revisão na revista Computers and Electronics in Agriculture (Qualis A2), já foram obtidos na forma de uma técnica, e respectivo aplicativo mobile, para medir a área das folhas que foi consumida por ataques de insetos. Um dos trabalhos de Mestrado está usando extração de características de imagens advindas do domínio educacional. O propósito é avaliar a qualidade dos repositórios de educação online considerando o conteúdo das ilustrações usadas no material veiculado. Este trabalho foi motivado pela experiência prévia do aluno que trabalha com o desenvolvimento de sistemas de gerenciamento de conteúdo online. Finalmente, um dos alunos de Mestrado está usando extração de características de texto para propor uma técnica de visualização capaz de identificar o contexto exploratório, baseando-se na semelhança de textos de notícias online, do texto vinculado aos itens de dados sendo visualizados. A ideia se baseia no princípio de que a completa análise de dados só ocorre quando amplo conhecimento contextualizado de domínio é usado no processo.
Capítulo 4. Conclusões
Referências Bibliográficas

Disponível em http://dx.doi.org/10.1109/TPAMI.2012.120


Disponível em http://doi.acm.org/10.1145/1735223.1735231


REFERÊNCIAS BIBLIOGRÁFICAS


Disponível em http://arxiv.org/abs/1112.5625


Disponível em http://arxiv.org/abs/1506.01406


Disponível em http://doi.acm.org/10.1145/2664551.2664579


REFERÊNCIAS BIBLIOGRÁFICAS


Identificação de fraudes
ORFEL: super-fast detection of defamation or illegitimate promotion in online recommendation

Gabriel Gimenes, Jose F Rodrigues Jr and Robson L F Cordeiro
University of Sao Paulo
Av Trab Sao-carlense, 400
Sao Carlos, SP,
Brazil-13566-590
{ggimenes,junio, robson}@icmc.usp.br

ABSTRACT
What if a successful company starts to receive a torrent of low-valued (one or two stars) recommendations in its mobile apps from multiple users within a short (say one month) period? Is it legitimate evidence that the apps have lost quality, or an intentional plan (via lockstep behavior) to steal market share through defamation? In case of a systematic attack to one’s reputation, it might not be possible to manually discern between legitimate and fraudulent interaction in the immense universe of possibilities of user-product recommendation. Previous works have focused on this issue, but none of them has considered the context, modeling, and scale that we work with in this paper. We propose one novel method named Online-Recommendation Fraud Excluder (ORFEL) to detect defamation and/or illegitimate promotion of online products using vertex-centric asynchronous parallel processing of bipartite (users-products) graphs. With an innovative algorithm, our results demonstrate efficacy – detecting over 95% of potential attacks; and efficiency – at least two orders of magnitude faster than the state-of-the-art. Over our new methodology, we introduce three contributions: (1) a new algorithmic solution; (2) a scalable approach; and (3) a novel context and modeling of the problem, which now addresses both defamation and illegitimate promotion. Our work deals with relevant issues of the Web 2.0, potentially augmenting the credibility of online recommendation to prevent losses to both customers and vendors.

Keywords
graphs, fraud detection, Web 2.0, data analysis

1. INTRODUCTION
In the Web 2.0, it is up to the users to provide content, including photos, text, and recommendations, among other user-generated information. In this universe, the more interaction (likes, recommendation, comments, ...) a product page or a user profile gets, the better are the potential profits that an individual or a company may achieve with automatic recommendation, advertisement, and/or priority in automatic search engines. In Google Play, for example, mobile apps depend, heavily, on high-valued (4 or 5 stars) recommendations in order to grow in importance and expand their pool of customers; on Amazon, users are offered the most recommended (better rated) products; and in TripAdvisor, users rely on other’s feedback to choose their next travels. The same holds for defamation, i.e., the act of lowering the rank of a given product by creating artificial, low-valued recommendations. Sadly, fraudulent interaction has come up in the Web 2.0 – fraudulent likes, recommendations, and evaluations define artificial interest that may illegitimately induce the importance of online competitors.

Attackers create illegitimate interaction by means of fake users, malware credential stealing, web robots, and/or social engineering. The identification of such behavior is of great importance for companies, not only because of the potential losses with fraud, but also because their customers tend to consider the reliability of a given website as an indicator of trust and quality. According to Facebook [8], fraudulent interaction is harmful to all users and to the Internet as a whole, being important to have true engagement of users around brands and content. However, catching up these kinds of attacks is a challenging task, especially when there are millions of users and millions of evaluated products defining billion-scale interaction daily. In such attacks, multiple fake users interact with multiple products at random moments [1] in a way that their behavior is camouflaged in the middle of million-per-second legitimate interaction. The core of the problem is: how to track temporal fraudulent user-product activity since the number of possible interactions is factorial? We want to find the so-called lockstep behavior – groups of users acting together, generally interacting with the same products at around the same time. As an example, imagine that an attacker creates a set of fake users to artificially promote his e-commerce website; then, he would like to comment and/or recommend his own webpages, posts, or advertisements to gain publicity that, fairly, should have come from real customers. Here, an attacker may refer to employees related to a given company, professionals (spammers) hired for this specific kind of job, web robots, or even anonymous users. The weak point of all these possibilities is that they must substantially interact with the attacked system within limited time windows; besides that, they must optimize their effort by using each fake user account to interact with multiple products. This behavior agrees with the lockstep definition, a pattern that is well-defined in many domains besides that of online recommendation, such as academic...
co-citation, social network interaction, and search-engine optimization. This is not a new problem, so we use in this paper the definition of lockstep behavior of Beutel et al. [4] (see Section 3.1).

The task of identifying locksteps is straightforwardly modeled as a graph problem in which we want to detect near-bipartite cores considering some given time constraint. The bipartite cores correspond to groups of users that interact with groups of products within limited time intervals. One lockstep can refer to defamatory, when the interactions refer to poor (low-valued) recommendations; or to illegitimate promotion, when the recommendations are high. As so, the problem generalizes to finding near-bipartite cores with edges whose weights correspond to the rank of the recommendations. It is important to note that we want to solve the problem without any previous knowledge about suspicious users, products, or moments when the frauds occurred.

This work goes deeper into the problem of lockstep identification, extending the state-of-the-art of previous solutions by means of the following contributions:

1. **Novel algorithmic paradigm**: we introduce the first vertex-centric algorithm able to spot lockstep behavior in Web-scale graphs using asynchronous parallel processing; vertex-centric processing is a promising paradigm that still lacks algorithms specifically tailored to its modus operandi.

2. **Scalability and accuracy**: we solve the problem for billion-scale graphs, in one single commodity machine, achieving efficiency that is comparable to that of cluster-based former works, whilst we achieve the same efficacy.

3. **Generality of scope**: we solve the problem for real weighted graphs ranging from social networks to e-commerce recommendation, extending the state-of-the-art of lockstep semantics to discriminate defamation and illegitimate promotion.

This paper follows a traditional organization: in Section 2, we review related works; in Section 3, we describe our proposal; in Section 4, we report our experimental results, including real data analyses; in Section 5 we discuss further applications; and in Section 6 we conclude the paper.

## 2. RELATED WORKS

### 2.1 Clustering

The identification of lockstep behavior refers to the problem of partitioning both the rows and columns of a matrix, which is known as co-clustering or bi-clustering. Some authors have worked on similar variations of the bi-clustering problem. For example, Papalexakis and Sidiopoulos used PARAFAC decomposition over the ENRON e-mail corpus [23]; Dhillon et al. used information theory over word-document matrices [7], and Banerjee et al. used Bregman divergence for predicting missing values and for compression [3]. Other applications include gene-microarray analysis, intrusion detection [22], natural language processing, collaborative filtering [15], and image, speech, and video analysis. Note that, in this problem setting, whenever time is considered in the form of time windows to be detected, we have a Non-deterministic Polynomial-time (NP-hard) problem [2] that prevents the identification of the best solution even for small datasets. In fact, we deal with issues fundamentally different from the problems proposed so far, which, according to Kriegel et al. [14], are not strictly comparable due to their specificities. Theoretically, our work resembles the works of Gupta and Ghosh [10] and of Crammer and Chechik [6]; similarly, we use local clustering principles but, differently, we are not dealing with one-class problems. Besides, the core of our technique is a variant of mean-shift clustering [5], now considering temporal and multi-dimensional aspects. As we mentioned before, our contribution relates not only to performance, but also to a novel algorithmic approach.

### 2.2 Web suspicious behavior detection

Many previous works use graph theory to detect suspicious behavior on the Web. This is the case of algorithm Crochet [24] that aims at identifying quasi-cliques based on an innovative heuristic; and also the case of MultiAspectForensics [19] that uses tensor decomposition to detect patterns within communities, including bipartite cores. In another work, Eigenspokes [25] uses singular-value decomposition to detect unexpected patterns in phone call data; meanwhile, Netprobe [21] uses belief propagation to find near bipartite cores in e-commerce graphs. However, none of the previous works focuses on performance at the same scale that we do; furthermore, these previous works do not consider the same problem with which we work, that is, the detection of a set of users fraudulently interacting with the same set of products at around the same time. In fact, the closest approach to our work is the CopyCatch algorithm [4], which solves the unweighted problem in a parallel, distributed setting – its experimental results reported considered one thousand machines. In this work we introduce a vertex-centric asynchronous parallel algorithm that works in one single commodity machine and whose performance rivals to what is reported in this former work, still achieving similar accuracy rates.

### 2.3 Vertex-centric graph processing

We use in this paper what is called vertex-centric processing [18]. Given a graph \( G = (V, E) \) in which the vertices are labeled from 1 to \( |V| \), we associate a value to each vertex and to each edge – for a given edge \( e = (u, v) \), \( u \) is the source and \( v \) is the target. With values associated to vertices and edges, vertex-centric processing corresponds to the graph scan approach depicted in Algorithm 1. The values are determined according to the computation that is desired, e.g., Pagerank or belief propagation; we illustrate it with hypothetical functions \( f \) and \( g \) in the algorithm. Evidently, a single scan is not enough for most useful computations, therefore the graph is scanned as many times as necessary for convergence. Graph processing, then, becomes what is defined in Algorithm 2.

#### Algorithm 1 Vertex-centric graph processing

```plaintext
procedure GRAPH_SCAN(GRAPH G)
  for \( i = 1 \) to \(|V|\) do
    set_ = set of edges adjacent to \( V[i] \)
    \( V[i].value \leftarrow f(set_); \)
    for each edge \( e \) in set_ do
      \( e.value \leftarrow g(V[i].value, e.value) \)
```

#### Algorithm 2 Graph processing

```plaintext
procedure GRAPH_PROCESSING
  while convergence criterion is not satisfied do
    GRAPH_SCAN(G)
```

The vertex-centric processing paradigm contrasts with usual graph traversal (breadth-first or depth-first) algorithms. While traversal-based algorithms support any kind of graph processing,
their strategy demands the graph to be entirely in memory, otherwise they would be prohibitively costly due to repeatedly random accesses to disk. On the other hand, the vertex-centric processing is limited to problems solved along the neighborhoods of the vertices (or with clever adaptations to such constraint); the good point is that it is adherent to disk-based processing, since it can suitably rely on sequential accesses to the disk. This kind of processing is not only prone to disk-based processing, but also to parallel processing according to which, each thread can be responsible for a different share of the vertices. This possibility yields to quite effective algorithms.

### 2.4 Asynchronous Parallel Processing

Many research groups have introduced systems devoted to large-scale graph processing based either on vertex-centric or edge-centric processing; this is the case of systems Pregel [18], Pegasus [13], PowerGraph [9], and GraphLab [17]. However, such works are parallel-distributed, and thus, they demand knowledge and management of costly computational clusters. More recently, a novel paradigm emerged in the form of frameworks that rely on asynchronous parallel processing, including systems GraphChi [15], TurboGraph [12], X-Stream [26] and MMap [27]. Such systems use disk I/O optimizations and the neighborhood information of nodes/edges in order to set up algorithms that can work in asynchronous parallel mode; that is, it is not required that their threads advance synchronously along the graph in order to reach useful computation. This approach has demonstrated successful to solve many problems, such as Pagerank, connected components, shortest path, and belief propagation, to name a few. Here we use vertex-centric graph processing over framework GraphChi; however, our algorithm can be adapted to any of the frameworks seen so far.

### 3. METHODOLOGY

#### 3.1 Problem formulation

In this section we provide a mathematical description of the lockstep problem. As mentioned in Section 1, we generalize the lockstep detection problem, amplifying its scope to defamation and illegitimate promotion. We do it by considering the weights of the edges, which correspond, for instance, to the numeric evaluation a user gave to a product on a recommendation website. Table 1 lists the symbols and definitions of our notation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$ and $n$</td>
<td>Number of nodes in each side of the bipartite graph.</td>
</tr>
<tr>
<td>$c$</td>
<td>Set of locksteps.</td>
</tr>
<tr>
<td>$M$</td>
<td>Set of lockstep ( c ).</td>
</tr>
<tr>
<td>$W$</td>
<td>( M \times N ) matrix holding the weight of each edge.</td>
</tr>
<tr>
<td>$N$</td>
<td>( M \times N ) matrix holding the timestamp of each edge.</td>
</tr>
<tr>
<td>$U[c]$ and $P[c]$</td>
<td>Set of users or products in lockstep ( c ).</td>
</tr>
<tr>
<td>$m$ and $n$</td>
<td>Minimum number of products and users in the lockstep to be considered valid.</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>Size of the timespan.</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Threshold percentage that the cardinality of the sets of products and users must satisfy to be in a lockstep.</td>
</tr>
<tr>
<td>$\lambda$ and $\kappa$</td>
<td>Function and threshold used to define defamation and promotion.</td>
</tr>
<tr>
<td>$v_p$</td>
<td>Current average time of suspicious recommendations to product $p$.</td>
</tr>
</tbody>
</table>

Table 1: Symbols and Definitions.

In this section we provide a mathematical description of the lockstep problem. As mentioned in Section 1, we generalize the lockstep detection problem, amplifying its scope to defamation and illegitimate promotion. We do it by considering the weights of the edges, which correspond, for instance, to the numeric evaluation a user gave to a product on a recommendation website. Table 1 lists the symbols and definitions of our notation.

Following we present the concept of a lockstep or, formally, a temporally-coherent near bipartite core.

### Definition 1

A set of products $P$ and a set of users $U$ comprise an \([n, m, \Delta t, \rho]\)-temporally-coherent near bipartite core if there exists $P_i \subseteq P$ for all $i \subseteq U$ such that:

\[
\begin{align*}
|P| & \geq m \\
|U| & \geq n \\
|P_i| & \geq \rho |P| \forall i \in U \\
(i, j) & \in c \ \forall i \in U, j \in P_i \\
\exists t_j & \in [t, t+\Delta t]. \ |f_j - L_{i,j}| \leq \Delta t \ \forall i \in U, j \in P_i
\end{align*}
\]

In other words, we have a suspicious lockstep if we find a set of products $P$ that was recommended by a set of users $U$ within a $\Delta t$ time window; we relax this definition with parameter $\rho$, which states that we also have a lockstep if we partially ($\rho$ percentage) satisfy this definition. Note that what makes the problem NP-hard, and challenging, is the temporal factor; also note that the problem refers to reducing the search space of frauds by pointing out suspiciousness, which can turn out to be actually fraudulent, or not.

Given this formulation, we consider the weight of the edges (recommendations) to define the concepts of defamation — shown in Equation 6, and illegitimate promotion — shown in Equation 7.

\[
W_{i,j} \leq \kappa \ \forall i \in U, j \in P_i
\]

Finally, we consider the problem as an optimization problem, whose objective is to catch as many suspect users as possible, while only growing $P$ until we satisfy parameter $m$. In Equation 8, we describe our objective function according to which we want to find $U[c]$ and $P[c]$ to maximize the number of users and their interactions for a given cluster $c$.

\[
\max_{U[c], P[c]} \sum_{c} q(I_{i,j}c|U[c]W_{i,j}|c, P[c])
\]

where

\[
q(u,w,w|P[c]) = \begin{cases} 
\sigma & \text{if } \sigma = \sum_{c} \phi(v_j, u_j) \lambda(w_j) \geq \rho m \\
0 & \text{otherwise}
\end{cases}
\]

\[
\phi(v_j, u_j) = \begin{cases} 
1 & \text{if } |v_j - t_j| \leq \Delta t \\
0 & \text{otherwise}
\end{cases}
\]

\[
\lambda(g_j) = \begin{cases} 
1 & \text{if } g_j \geq \kappa \text{ for promotion} \\
0 & \text{otherwise}
\end{cases}
\]

\[
\lambda(g_j) = \begin{cases} 
1 & \text{if } g_j \leq \kappa \text{ for defamation} \\
0 & \text{otherwise}
\end{cases}
\]

Equations 11 and 12 refer to our definitions of illegitimate promotion and defamation, while Equation 9 shows how we incorporate these weight constraints in the general problem, through the definition of a threshold function $\lambda$.

### 3.2 ORFEL algorithm

In order to find locksteps, we propose the Online-Recommendation Fraud ExCluder (ORFEL), one novel, iterative algorithm that follows the idea of vertex-centric processing introduced in Section 2.3. Each iteration executes two functions: \textit{updateProducts} and \textit{updateUsers} that, respectively, will add/remove products and users from a lockstep that is being constructed. The algorithm iterates until convergence – that
is, when sets $P$ and $U$ stabilize for all the locksteps that were identified, presenting insignificant changes between iterations. The full pseudo-code of ORFEL is in Algorithm 3.

Initialization

The algorithm relies on seeds to search the data space; the idea is to have each seed inspecting its surroundings looking for one local maximum. One seed in the algorithm corresponds to one potential lockstep, which comprises a set of products and a set of users. Therefore, the initial seeds correspond to minimum locksteps, that is, one single product, and a few ($\geq 1$) users – the only requirement to the initial set of users is that it cannot be an empty set. The initialization randomly chooses products of the dataset, each one corresponding to one seed. Then, for each product (seed), ORFEL randomly chooses a constant small number of users that recommended this product. This is necessary so that the algorithm has initial elements – $P[i]$ and $U[i]$ for every $i$th-lockstep – scattered throughout the search space. After setting up these initial locksteps the algorithm will iteratively have them grow in number of products and users.

Product update

In the `updateProducts` function we only consider vertices that are products; what means that changes are made only in set $P[i]$. This function is called for every product testing if this product fits in one of the lockstep; the test is performed for all locksteps. One product enters a given lockstep if at least $\rho$ percent of the users currently in the lockstep recommend that product within a $\Delta t$ time window. Also, the algorithm only considers recommendations that fit the given weight constraint (represented by the $\lambda$ function) when accounting for the percent of users that recommend the candidate product, characterizing it as either defamation or illegitimate promotion.

Additionally, when a lockstep already has $m$ products, that is, it has already achieved the maximum number of products per lockstep, we test if it is worth to swap a product currently in the lockstep for the candidate product. This test is similar to the one used to add a product, except that, to be swapped, now the candidate product must contain a superset of the set of recommendations that the current product has. This is an heuristic approach that leads to an additional coverage of the search space because, as we look for supersets of recommendations, the locksteps tend to increase in size.

User update

In turn, function `updateUsers` considers only vertices that are users; changes are made only to set $U[i]$. Similarly to what is done in the `updateProducts` step, we update each lockstep separately by testing if the current user can be added to it. A candidate user will enter a lockstep if it recommends at least $\rho$ percent of the products in the cluster within a $2\Delta t$ time window of each of the products’ time centers - that is, the average recommendation time on that product inside the lockstep - and if it fits the desired weight constraint. If the candidate user fills the requirement it is added to that lockstep. It is important to note that in this step we allow users that are outside of the actual $\Delta t$ time window (we propose to use $2\Delta t$, following the empirical evidence of both our work and the state-of-the-art approach) to enter the lockstep; this is the mechanism that drives the lockstep towards a better local maximum, in the case that there exists one.

End iteration

Finally, once the `updateUsers` step is complete, we must run function `endIteration`. In this step, for all locksteps, the algorithm sorts the $2\Delta t$ recommendations by their time stamp and scans this set sequentially looking for the subset that maximizes the recommendation criterion (number of recommendations); the desired subset must fit a $\Delta t$ window. This is the core mechanism of the algorithm; what it does is to let a $2\Delta t$ time window to take place first, then, from the corresponding set of recommendations, it selects a subset that maximizes the desired criterion. This mechanism is what makes the seeds “inspect” their $2\Delta t$ neighborhoods. If the recommendation set changes, a new iteration will lend new products and users to enter/swap into the lockstep, until convergence. Once a seed finds a local maximum, it gets stuck and does not change anymore.

Note that some seeds will converge sooner than others, getting stuck to locksteps smaller than the $m$ and $n$ parameters. These seeds are considered dead (no iterations between iterations) and are ignored by the algorithm. This occurs right after the second iteration, after which the number of locksteps (live seeds) becomes smaller than the number of initial seeds. The algorithm converges when all the seeds are dead.

3.3 Discussion about the parameters

As shown in Algorithm 3, ORFEL has five parameters: $m$, $n$, $\rho$, $\Delta t$ and $nSeeds$. The first two, $m$ and $n$, refer, respectively, to the cardinality of products and users that the algorithm must verify when evaluating suspicious locksteps. Parameter $\rho$ is the acceptable percentage (the tolerance fraction) of products (\rho \times m) for the algorithm to state that a bipartition is in fact suspicious. Although one could change the $\rho$ value, we suggest using no less than 80%, otherwise the locksteps could degenerate. Parameter $\Delta t$ defines the time window within which the interactions (recommendations) should take place. Parameter $nSeeds$ refers to the number of seeds that the algorithm will spread through the data universe, each one searching for one suspicious lockstep.

Parameters $m$ and $n$ define the aspects of the suspect behavior that we are considering. Raising the value of $m$ or $n$ means that we want to search for suspect behaviors involving more products and/or users. Talking about $m$ and $n$ relates to what we call “AttackSize”, that is, the dimensions of the potential attacks that we presume to exist. These parameters filter out attacks that are too small and/or too big, what may be desired depending on the domain. In the section of experiments, we evaluate how different configurations of AttackSize impact the efficacy of our algorithm. Parameter $\rho$ makes the algorithm flexible to different sizes of attacks, including attacks in which the users attack only a fraction (percentage) of the expected number of products, $m$, in the locksteps. In practice parameters $\rho$ informs the algorithm to have a tolerance around $m$ and $n$. These parameters, along with $\Delta t$, depend on the semantics of the problem’s domain.

Parameter $\Delta t$ defines the timespan desired by the analyst when searching for attacks. For instance, let’s say we are searching for attacks in a social network; we could argue that a timespan as big as a couple hours is enough to find ill-intended interactions. On the other hand, if we are talking about online reviews, one could think of one week. The same reasoning can be used to define parameters $m$ and $n$.

Finally, parameter $nSeeds$ controls ORFEL’s potential of discovery; as we demonstrate in the experiments, the bigger the data, the bigger must be the number of seeds following a linear correlation.

3.4 Convergence

Our algorithm focuses on finding a set of local maxima for the objective function defined in Equation 8, which is bounded because there is a limited set of users and products. With a bounded func-
Algorithm 3 ORFEL Algorithm.

function ORFEL(n, m, p, \Delta, nSeeds)
    Initialize U[nSeeds], P[nSeeds] \triangleright Initial Seeding
    repeat
        U' = U
        P' = P
        for each product p in |V| do
            P = updateProducts(p)
        for each user u in |V| do
            U = updateUsers(u)
        endIteration()
    until U' = U and P' = P
    return [U, P]

procedure UPDATEPRODUCTS(vertex)
    for each Lockstep c ∈ C do
        Recomms ← U[c].edges ∩ vertex.edges
        timeCenter ← avgtime(Recomms)
        for each edge e in Recomms do
            if |time - timeCenter| > \Delta and \lambda(e.weight) then
                Recomms = Recomms - {e}
            if |P[e]| < m then
                if ([Recomms] ∩ [U[c]]) ≥ p then
                    P[c] = P[c] ∪ {vertex}
            else
                for each product p ∈ P[c] do
                    if p.Recomms ⊂ Recomms then
                        swap = p
                        P[c] = (P[c] - {swap}) ∪ {vertex}
        endIteration()
    for each Lockstep c ∈ C do
        Recomms ← P[c].edges ∩ vertex.edges
        pCenter ← avgtime(u, e.vertex, u ∈ U[c])
        if |e.time - pCenter| > \Delta and \lambda(e.weight) then
            Recomms = Recomms - {e}
        if ([Recomms] ∩ [P[c]]) ≥ \rho then
            U[c] = U[c] ∪ {vertex}

procedure UPDATEUSERS(vertex)
    for each Lockstep c ∈ C do
        Recomms = P[c].edges ∩ vertex.edges
        for each edge e in Recomms do
            pCenter = avgtime(u, e.vertex, u ∈ U[c])
            if |e.time - pCenter| > \Delta and \lambda(e.weight) then
                Recomms = Recomms - {e}
        if ([Recomms] ∩ [P[c]]) ≥ \rho then
            U[c] = U[c] ∪ {vertex}

procedure ENDITERATION
    for each Cluster c ∈ C do
        for each product p ∈ P[c] do
            Sort U[c] by the time of the Recomms
            Scan sorted U[c] for the 2\Delta-subset that maximizes
            the number of Recomms
            Remove the users from U[c] that are not in the subset

Table 2: Datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Nodes</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon.FineFoods</td>
<td>330,000</td>
<td>550,000</td>
</tr>
<tr>
<td>Amazon.Movies</td>
<td>1,140,000</td>
<td>8,000,000</td>
</tr>
<tr>
<td>Synthetic.C</td>
<td>10,000,000</td>
<td>100,000,000</td>
</tr>
</tbody>
</table>

4. EXPERIMENTS

4.1 Experimental setting

We implemented ORFEL using Java 1.7 over the GraphChi platform as stated in Section 2.4. We ran our experiments on an i7-4770 machine with 16 GB of RAM, and 2TB 7200RPM HDD; for the SSD tests, we used a 240GB drive with I/O at 450MB/s. For full reproducibility, the complete experimental set is available at www.icme.uap.br/pessoa/junio/ORFEL/index.htm, including source codes and graph/lockstep generators.

As for the datasets\(^1\), we considered two real world graphs: Amazon.FineFoods and Amazon.Movies. Both datasets comprise user-product recommendation data from the Amazon website; the first one refers to the fine foods product section and the second one contains reviews of movies. Also, we had the timestamp of each review and its numeric evaluation ranging from 1 to 5. We also used synthetic datasets so to generalize the scope of our tests. We use a bipartite graph generator that works based on the G\(_{nmk}\) model available on NetworkX \([11]\), in which \(n\) stands for the number of nodes in the first bipartite set; \(m\) stands for the number of nodes in the second bipartite set; and \(k\) is the number of randomly generated edges connecting both sets. Table 2 lists the two Amazon datasets plus the synthetic dataset Synthetic.C, which was generated using \(n = 2,000,000, m = 8,000,000\) and \(k = 100,000,000\). Additionally, we generated benchmark datasets that are larger versions of dataset Synthetic.C; they were used for scalability tests, as we describe in Section 4.4.

Experimental goals

The main feature expected from ORFEL is the ability to detect lockstep attacks, either of defamation or of illegitimate promotion. As mentioned in Section 2.1, this is an NP-hard problem that we approximately solve via an optimization approach. Considering these aspects, we verify the correctness in Section 4.2, the efficacy (i.e.,

\(^1\) Available at the Snap project \([16]\) web page at http://snap.stanford.edu/.
the ability to find the majority (> 95%) of the lockstep attacks) in Section 4.3, the efficiency (i.e., the ability to reach efficacy within desired time constraints) in Section 4.4, and the computational cost of the whole approach in Section 4.5.

4.2 Initial tests under controlled conditions

In the first experiment, we used 4 small (thousand-edge scale) synthetic graphs to verify that the algorithm would detect locksteps only when they were really there. These are the controlled conditions of our experimentation, that is, we wanted to make sure that the algorithm would not point out suspect behaviors when we had ascertained that there were none to be detected. We generated synthetic graphs in which no products nor sets of products would configure a given suspect behavior, such as 10 pages and 25 users. We then ran ORFEL with varying parameters, including \( m = 5, n = 10 \) and \( m = 10, n = 25 \) and verified that, as expected, no lockstep was detected. The same results could be inferred from the algorithm description of Section 3.2 and from the discussion carried out in Section 3.4, still, we verified this feature empirically.

4.3 Efficacy

We define efficacy as the ability of finding the majority (> 95%) of the locksteps. To test this feature, we created controlled conditions with artificial attacks appended to our datasets that permitted us to evaluate the output of the algorithm. Algorithm 4 shows how we generated such attacks, by randomly choosing a group of pages and users and then connecting them within a limited \( \Delta t \) time window. This was necessary because, since the problem is NP-hard, we would not be able to know whether the output of the algorithm is correct considering uncontrolled conditions. The problem is a variation of subspace clustering, considering the semantic that the clusters (locksteps) are unusual and, therefore, suspect. Note that we did not focus on the issue of determining whether a given suspect lockstep is actually an attack, since this is one distinct problem that demands extra information (identification, customer profile, and so on) to be evaluated by means of false-positive and true-positive rates.

**Attack size**

In order to analyze the ability of the algorithm to find locksteps with different sizes, we ran experiments for each of the three datasets described in Table 2. We fixed \( m \) and \( n \) in each case while varying the sizes of the artificial attacks appended to the dataset, so that we would be able to see how effective the algorithm is, depending on the size of the attacks.

In the first experiment, we set \( n \) to 10 and \( m \) to 5 and introduced the size of the attacks.

**Algorithm 4: Lockstep Generator**

```python
procedure LOCKSTEPPER(Graph G, nUsers, nPages, \( \Delta t \))
    users = GetRandomUsers(G, nUsers);
    pages = GetRandomPages(G, nPages);
    for each Page P ∈ Pages do
        timestamp = getRandomTimeStamp();
        rating = getRandomRating();
        for each User U ∈ Users do
            rating = getRandomRating();
            addEdge(G, U, P, newTimestamp, rating);
```

in each dataset artificial attacks with sizes varying from 10 users and 5 products to 120 users and 60 products. This allowed us to observe the percentage of attacks caught for each configuration. In Figure 1, one can see that the algorithm was able to detect over 95% of the attacking users for different attack sizes and different parameters. Note that we varied parameters \( n \) and \( m \) to detect more attacks of each specific size with the goal of understanding how the choice of parameters works against different attack sizes. One can see that each graphic followed a similar behavior, indicating that the percentage of users caught tends to grow with the size of the attacks, that is, the bigger the attacks are, the more likely we are to detect them with a given configuration, which intuitively makes sense.

**Number of seeds**

We ran the algorithm over our three test datasets; we did it 4 times for each one – as the algorithm is non-deterministic – and calculated the average response. It was a requirement that none of the 4 runs would present discrepant results; we were able to verify that since the variance of the results was at the order of 1%. We introduced 20 synthetic attacks (10 defamations and 10 illegitimate promotions) in each dataset and varied the number of seeds from 1,000 to 7,000. In Figure 2, it is possible to see that for the smaller dataset (550 K edges) – the Amazon.FineFoods, we were able to catch over 95% of the attacks with 4,000 seeds. Interestingly, the average of attacks caught with 5,000 seeds was significantly lower, what indicates that the algorithm reached its peak performance around 4,000 seeds and only had some variation afterwards due to its non-determinism. In Figure 2, we see that the algorithm caught over 95% of the attacks with 6,000 seeds in a dataset with 8 M edges (Amazon.Movies); and, in Figure 2, it caught over 95%
of the attacks with 7,000 seeds in a dataset with 100 M edges \( \text{Synthetic.C} \). We conclude that the number of seeds to reach high rates of detection goes with the data size, with a number approximate to \( 10^3 \times \log(\text{number of edges}) \), which for our 3 datasets would be approximately: 5800, 6900 and 8000, respectively.

![Figure 2: Experiments of efficacy: the percentage of attacks caught versus the number of seeds. Efficacy is demonstrated when over 95% of the attacks are caught. Parameters \([n,m,p,\text{AttackSize|Users|Products}]\) are: \text{Synthetic.C} \([50,25,0.8,750,375])\); Amazon.Movies \([50,25,0.8,500,250])\); Amazon.FineFoods \([10,5,0.8,50,25])\).](image)

From this experiment, we were able to verify that \text{ORFEL} is effective; we were able to identify more than 95% of the attacks in three datasets of different sizes. Also notice that the algorithm is flexible in relation to the attack sizes, as observed in the parameters of Figure 2. It means that \text{ORFEL} fits the peculiarities of different domains.

### Real data experimentation

We performed additional experiments in both the real datasets Amazon.Movies and Amazon.FineFoods \emph{without the inclusion of any synthetic data}. For this context, we considered that a suspect behavior would be 20 users positively recommending 6 movies or food products in less than a week, which, in this semantic context, is an intense load of recommendation. We ran the algorithm and found 37 suspect attacks; we took 8 minutes to achieve convergence for the larger dataset (Amazon.Movies). Since the execution time was quite small, we were able to try variations of the desired attacks, with 15 users and 7 movies within a week, and also 10 users and 10 movies within three days. After manually analyzing the suspect locksteps, we discovered that they were caused by Amazon’s policy of using different identification numbers for different flavors/sizes of the same food product, and for different versions of the same movie, while merging their reviews. Although these were not actual attacks, this simple experiment revealed a behavior that should be better analyzed since the Amazon’s policy could eventually lead customers to a misleading choice. In summary, we were able to identify, in a universe of 1,140,000 nodes and 8,000,000 edges, really small temporal patterns that demanded attention. We also note the reduced time to accomplish such results, which allowed us to test on different parameters according to the semantics of the dataset.

### 4.4 Efficiency and scalability

Given the current scale of network-like data, our method is supposed to be efficient, that is, it must handle billion-scale graphs and finish in reasonable time. We test this requirement with synthetic, benchmark datasets that are larger versions of dataset \text{Synthetic.C}. Although there are plenty of real data related to our problem (network data including edge weights and time stamps), such data is usually not shared by companies due to privacy matters.

#### Preprocessing

Asynchronous Parallel Processing platforms like those reviewed in Section 2.4 demand a preprocessing step in which the data is organized and formatted in accordance to the platform’s paradigm. In our case, the preprocessing step converts text to binary data, then it sorts and writes the vertices so to have them read from disk with sequential scans, minimizing the number of seeks. In wall-clock time we take nearly 45 min. to preprocess 1 billion edges on a mechanical disk, and nearly 15 min. on a solid-state disk – for a given dataset, preprocessing is necessary only once, no matter how many times we shall process the data later on.

#### Number of edges

First we test on the number of edges of the graph as, for our algorithm, the data size is the main constraint for time scalability. In the first experiment, we ran \text{ORFEL} with 100 seeds; we took 7 \( \approx 3.23 \) hour) over a solid-state disk. We argue that this performance is \emph{super-fast} because the previous work (Beutel et al. \[4\] – Fig 4a) took \( \approx 0.5 \) hour to do a similar processing with one thousand machines over MapReduce, while we used one \emph{single} commodity machine. The difference of performance is because the former work executes a sequential (non-parallel) algorithm that needs to compute one seed at a time in each machine; therefore, performance came at the cost of thousand machines, each executing one instance of the computation, in a distributed environment with heavy communication demands. Differently, the design of our algorithm explored the fact that the problem can be solved considering only the neighborhood of each node, what allowed us to process the graph in parallel asynchronous mode, with multiple seeds being solved simultaneously. The performance gains are not only remarkable, they also made it possible to solve the problem in commodity hardware.

#### Number of seeds

We also test the runtime of the algorithm in relation to the number of seeds. We ran \text{ORFEL} over 100 million edges varying the number of seeds from 100 to 5,000. Figure 4 shows the plot for this experiment; in the figure one can see that, also in this case, the runtime scales linearly with the number of seeds. The runtime for 100 seeds is 10 min., while for 5,000 seeds it takes 298 min.; that is a 30-times increase in time for a 50-times increase in the problem input; or a slope smaller than 1 for the relationship runtime \( \times \) number of seeds, just as desired.

#### 4.5 Computational cost

Given a graph \( G \) of size \( D \) bytes, a main memory of size \( M \) bytes, and disk blocks of size \( b \) bytes; \text{ORFEL} splits the graph into \( |P| = D/M \) parts. Each part contains edges that are sorted in disk according to their source vertices so that the graph is processed by reading the parts once as targets and then as sources. Therefore, in order to read the entire graph, it is necessary to read \( B = D/b \) disk blocks twice, or \( 2B \) times. Besides that, for each
Figure 3: Experiments of scalability on the number of edges: linear growth of computation time (100 seeds) versus the number of edges for mechanical disk (HDD) and for solid-state disk (SSD). Parameters $[n,m,\rho,\text{AttackSize(Users,Products)},n\text{Seeds}]$ are: $[50,25,0.8,(500,250),100]$.

Figure 4: Experiments of scalability on the number of seeds: linear growth (line coefficient < 1) of computation time versus the number of seeds for mechanical disk (HDD) over 100 million edges. Parameters $[n,m,\rho,\text{AttackSize(Users,Products)}]$ are: $[50,25,0.8,\{500,250\}]$.

part it is necessary to read the other $P-1$ parts, what leads to $P^2$ disk seeks. It brings us to the total disk operation as given by $P^2$ disk seeks + $2B$ block reads per iteration.

Also, considering the ORFEL algorithm, we can see that it runs for $I$ iterations. In each iteration, besides the disk operation, it runs once for each of the $S$ seeds (worst case); each time, it processes all the $|E|$ edges of the graph, which are in memory. Therefore, the processing cost of the algorithm is $I*O(S*|E|)$.

Each iteration of the algorithm asks for a re-organization step in which the locksteps of each seed are redefined based on the results annotated at the last iteration. For $I$ iterations, $S$ seeds, $n$ users and $m$ products, this step runs at cost $I*O(S*n*(m+\log(m)))$, part of this cost is due to the operation of sorting in memory (in logarithmic time). This is the cost of the worst case, when the algorithm processes all the seeds – this number drops abruptly after a few iterations because the majority of the seeds does not grow; instead, they get stuck to a local maximum that is too small to be considered a lockstep, being dismissed for further iterations.

Finally, the total cost of ORFEL is $I*(P^2 \text{ disk seeks} + 2B \text{ block reads} + O(S*|E|) + O(S*n*m*\log(m)))$. Since the cost of processing is 6 orders of magnitude smaller than that of a mechanical disk and 4 orders smaller than that of a solid-state-disk, the processing cost is insignificant. As so, the computational cost of ORFEL is $I*(P^2 \text{ disk seeks} + 2B \text{ block reads})$. Note, from our analysis, that all the cost depends on the available memory, which is used as a buffer for data coming from disk; hence, all the time measurements reported in this section could be smaller if we had used more memory.

5. DISCUSSION AND FURTHER APPLICATIONS

As previously mentioned, ORFEL suits for many other problems that can be represented as graphs, lending support to further applications.

5.1 Social networks

In the case of social networks, the usual interaction is to like a given post (as for Google+ or Facebook); for this configuration, locksteps characterize solely by illegitimate promotion, when a given post (or page) gets fake likes from attackers willing to make it more relevant than it really is. According to the model of ORFEL, this problem refers to a bipartite graph in which the edges are not weighted (weight 1 for all the edges).

5.2 Journal co-citations

Given the pressure for relevance and impact, some scientific journals may use a co-citation scheme in which one journal cites the other and vice-versa, just like in the case spotted by Nature in 2013 [20]. According to this scheme, a variant of lockstep behavior, one journal tends to favor papers citing a specific other journal; and even to have editors recommend authors what to cite in their work as a condition for publication.

Identifying this kind of lockstep behavior is not trivial because systematic co-citation tends to disappear along years of publications, when such schemes are covered by the volume of legitimate citations and by the time magnitude – it is reasonable to have co-citation between any two journals in periods of 10 years, for example. The problem becomes even harder if more than two journals – three or four – set up the scheme, when simple journal-to-journal interaction may not be sufficient to detect a scheme. In other words, the temporal factor and the volume of data make it a problem much harder than simply detecting bipartite sub graphs. This problem is another instance of the lockstep detection treated in this work. With our algorithm it is possible to spot co-citation occurring, let’s say, within periods of 1 or 2 years for any number of journals. For time intervals such as those, one may suspect if a set of journals cite each other with high intensity.

For this specific case, our problem model changes a little. In our model, we assumed users recommending products – a bipartite graph; for detecting journal co-citations we must have the set of journals under investigation replicated, that is one given journal appears in the model twice: once as a citing journal and once as a cited journal, what also defines a bipartite graph as expected by our algorithm. The output of our algorithm, then, shall present sub bipartite graphs. However, different from the user-product model, it is not enough to identify sub bipartite graphs as an indication of fraud; it is also necessary to have a high similarity between the two sets of the nodes of the bipartite sub graph. This is straightforwardly achieved with the Jaccard set similarity $Jaccard = \frac{|set_1 \cap set_2|}{|set_1 \cup set_2|}$, which returns 1 if two sets are exactly the same, and 0 if they are totally different. For the co-citation problem, our algorithm could
be set to return the set of bipartite sub graphs ordered by their Jaccard similarity. Of course, our algorithm spots cases that are just suspicious, but that are not definitive frauds; these cases must go through human interpretation because it is expected that the journals with the highest impacts cite each other, however this is not expected in journals with lower impact rates. Our algorithm cannot only detect suspicious co-citation cases, it can do it with high scalability. Given our processing performance, our algorithm can virtually inspect all the publication interaction ever produced in just a few hours.

6. CONCLUSIONS

We treated the problem of systematic attacks to defame or to promote entities that define bipartite networks, such as in the case of user-product recommendations, user-apps evaluations, or journal-journal co-citations. Such attacks are performed by means of fake users, malware credential stealing, web robots, and/or social engineering. The problem is modeled as a bipartite weighted and timestamped graph in which we want to detect temporal near-bipartite cores. The challenge here is known to be NP-hard, so that we introduced an approximate, optimization-based algorithm to solve the problem; our approach is based on vertex-centric asynchronous parallel processing. Our main contributions are:

1. **Novel algorithmic paradigm:** we developed the ORFEL algorithm along with a formalization of its fundamentals, which include an optimization approach over a weighted and timestamped graph. To the best of our knowledge, ORFEL is the first vertex-centric algorithm able to spot lockstep behavior in Web-scale graphs using asynchronous parallel processing;

2. **Scalability and accuracy:** we demonstrated empirical efficacy (over 95% of detection) and superior efficiency – with a performance, in one single commodity machine, that rivals that of previous cluster-based works with similar accuracy at the scale of billion edges;

3. **Generality of scope:** we were able to detect both defamation and illegitimate promotion according to a novel modeling of the problem, expanding the scope of previous works to high-impact applications, like e-commerce, scientific communities, and social networks.

Finally, we emphasize the importance of detecting lockstep behavior, either for defamation or promotion, because these frauds can harm both customers and vendors, inducing the trading of unverified products. We show the applicability of our approach in real graphs from e-commerce. We also note that the relevance of this problem is growing as the Web 2.0 is becoming popular, in which online trading and recommendation play a central role in users’ habits.

7. REFERENCES


Processamento massivo de grafos
M-Flash: Fast Billion-scale Graph Computation Using a Bimodal Block Processing Model

Hugo Gualdrón
University of Sao Paulo
Sao Carlos, SP, Brazil
gualdron@icmc.usp.br

Robson L. F. Cordeiro
University of Sao Paulo
Sao Carlos, SP, Brazil
robson@icmc.usp.br

Jose F. Rodrigues Jr.
University of Sao Paulo
Sao Carlos, SP, Brazil
junio@icmc.usp.br

Duen Horng (Polo) Chau
Georgia Institute of Technology
Atlanta, USA
polo@gatech.edu

Minsuk Kahng
Georgia Institute of Technology
Atlanta, USA
kahng@gatech.edu

U Kang
KAIST
Daejeon, Republic of Korea
ukang@kaist.ac.kr

ABSTRACT
Recent graph computation approaches such as GraphChi, X-Stream, TurboGraph and MMap demonstrated that a single PC can perform efficient computation on billion-scale graphs. While they use different techniques to achieve scalability through optimizing I/O operations, such optimization often does not fully exploit the capabilities of modern hard drives. Our main contributions are: (1) we propose a novel and scalable graph computation framework called M-Flash that uses a new, bimodal block processing strategy (BBP) to boost computation speed by minimizing I/O cost; (2) M-Flash includes a flexible and deliberately simple programming model that enables us to easily implement popular and essential graph algorithms, including the first single-machine billion-scale eigensolver; and (3) we performed extensive experiments on real graphs with up to 6.6 billion edges, demonstrating M-Flash’s consistent and significant speed-up over state-of-the-art approaches.

Keywords
graph algorithms, single machine scalable graph computation, Bimodal Block Processing model

1. INTRODUCTION
Large graphs with billions of nodes and edges are increasingly common in many domains and applications, such as in studies of social networks, transportation route networks, citation networks, and many others. Distributed frameworks have become popular choices for analyzing these large graphs (e.g., GraphLab [5], PEGASUS [8] and Pregel [14]). However, distributed approaches may not always be the best option, because they can be expensive to build [12], hard to maintain and optimize.

These potential challenges prompted researchers to create single-machine, billion-scale graph computation frameworks that are well-suited to essential graph algorithms, such as eigensolver, PageRank, connected components and many others. Examples are GraphChi [12] and TurboGraph [6]. Frameworks in this category define sophisticated processing schemes to overcome challenges induced by limited main memory and poor locality of memory access observed in many graph algorithms [16]. For example, most frameworks use an iterative, vertex-centric programming model to implement algorithms: in each iteration, a scatter step first propagates the data or information associated with vertices (e.g., node degrees) to their neighbors, followed by a gather step, where a vertex accumulates incoming updates from its neighbors to recalculate its own vertex data.

Recently, X-Stream [18] introduced a related edge-centric, scatter-gather processing scheme that achieved better performance over the vertex-centric approaches, by favoring sequential disk access over unordered data, instead of favoring random access over ordered and indexed data (as it occurs in most other approaches). When studying this and other approaches [13][12], we noticed that despite their sophisticated schemes and novel programming models, they often do not optimize for disk operations, which is the core of performance in graph processing frameworks. For example, reading or writing to disk is often performed at a lower speed than the disk supports; or, reading from disk is commonly executed more times than it is necessary, what could be avoided.

In the context of single-node, billion-scale graph processing frameworks, we present M-Flash, a novel scalable framework that overcomes many of the critical issues of the existing approaches. M-Flash outperforms the state-of-the-art approaches in large graph computation, being many times faster than the others. More specifically, our contributions include:

1. M-Flash Framework & Methodology: we propose the novel M-Flash framework that achieves fast and scalable graph computation via our new bimodal block model that significantly boosts computation speed and reduces disk accesses by dividing a graph and its node data into blocks (dense and sparse), thus minimizing the cost of I/O. Complete source-code of M-Flash is re-
2. Programming Model: M-Flash provides a flexible, and deliberately simple programming model, made possible by our new bimodal block processing strategy. We demonstrate how popular, essential graph algorithms may be easily implemented (e.g., PageRank, connected components, the first single-machine eigensolver over billion-node graphs, etc.), and how a number of others can be supported.

3. Extensive Experimental Evaluation: we compared M-Flash with state-of-the-art frameworks using large real graphs, the largest one having 6.6 billion edges (YahooWeb [22]). M-Flash was consistently and significantly faster than GraphChi [12], X-Stream [18], TurboGraph [6] and MMap [13] across all graph sizes. And it sustained high speed even when memory was severely constrained (e.g., 6.4X faster than X-Stream, when using 4GB of RAM).

2. RELATED WORK

A typical approach to scalable graph processing is to develop a distributed framework. This is the case of PEGASUS [8], Apache Giraph (http://giraph.apache.org/), Powergraph [5], and Pregel [14]. Differently, in this work, we aim to scale up by maximizing what a single machine can do, which is considerably cheaper and easier to manage. Single-node processing solutions have recently reached comparative performance to distributed systems for similar tasks [9].

Among the existing works designed for single-node processing, some of them are restricted to SSDs. These works rely on the remarkable low-latency and improved I/O of SSDs compared to magnetic disks. This is the case of TurboGraph [6] and RASP [23], which rely on random accesses to the edges — not well supported over magnetic disks. Our proposal, M-Flash, avoids this drawback at the same time that it demonstrates better performance over TurboGraph.

GraphChi [12] was one of the first single-node approaches to avoid random disk/edge accesses, improving the performance for mechanical disks. GraphChi partitions the graph on disk into units called shards, requiring a preprocessing step to sort the data by source vertex. GraphChi uses a vertex-centric approach that requires a shard to fit entirely in memory, including both the vertices in the shard and all their edges (in and out). As we demonstrate, this fact makes GraphChi less efficient when compared to our work. Our M-Flash requires only a subset of the vertex data to be stored in memory.

MMap [13] introduced an interesting approach based on OS-supported mapping of disk data into memory (virtual memory). It allows graph data to be accessed as if they were stored in unlimited memory, avoiding the need to manage data buffering. This enables high performance with minimal code. Inspired by MMap, our framework uses memory-mapping when processing edge blocks but, with an improved engineering, our M-Flash consistently outperforms MMap, as we demonstrate.

Our M-Flash also draws inspiration from the edge streaming approach introduced by X-Stream’s processing model [18], improving it with fewer disk writes for dense regions of the graph. Edge streaming is a sort of stream processing referring to unrestricted data flows over a bounded amount of buffering. As we demonstrate, this leads to optimized data transfer by means of less I/O and more processing per data transfer.

3. M-Flash

In this section, we first describe how a graph is represented in M-Flash (Subsection 3.1). Then, we detail how our block-based processing model enables fast computation while using little RAM (Subsection 3.2). Subsection 3.3 explains how graph algorithms can be implemented using M-Flash’s generic programming model, taking as examples well-known, essential algorithms. Finally, system design and implementation are discussed in Section 3.4.

The design of M-Flash considers the fact that real graphs have varying density of edges; that is, a given graph contains dense regions with much more edges than other regions that are sparse. In the development of M-Flash, and through experimentation with existing works, we noticed that these dense and sparse regions could not be processed in the same way. We also noticed that this was the reason why existing works failed to achieve superior performance. To cope with this issue, we designed M-Flash to work according to two distinct processing schemes: Dense Block Processing (DBP) and Streaming Partition Processing (SPP). Hence, for full performance, M-Flash uses a theoretical I/O cost based optimization scheme to decide the kind of processing to use in face of a given block, which can be dense or sparse. The final approach, which combines DBP and SPP, was named Bimodal Block Processing (BBP).

3.1 Graphs Representation in M-Flash

A graph in M-Flash is a directed graph $G = (V, E)$ with vertices $v \in V$ labeled with integers from $1$ to $|V|$, and edges $e = (source, destination), e \in E$. Each vertex has a set of

Copyright the Authors
**Blocks in M-Flash:** Given a graph \( G \), we divide its vertices \( V \) into \( \beta \) intervals denoted by \( I^{(p)} \), where \( 1 \leq p \leq \beta \). Note that \( I^{(p)} \cap I^{(p')} = \emptyset \) for \( p \neq p' \), and \( \bigcup_p I^{(p)} = V \). Thus, as shown in Figure 1, the edges are divided into \( \beta^2 \) blocks. Each block \( G^{(p,q)} \) has a source node interval \( p \) and a destination node interval \( q \), where \( 1 \leq p,q \leq \beta \). In Figure 1, for example, \( G^{(2,3)} \) is the block that contains edges with source vertices in the interval \( I^{(2)} \) and destination vertices in the interval \( I^{(3)} \). In total, we have \( \beta^2 \) blocks. We call this on-disk organization of the graph as partitioning. Since M-Flash works by alternating one entire block in memory for each running thread, the value of \( \beta \) is automatically determined by equation:

\[
\beta = \left\lfloor \frac{2\varphi T |V|}{M} \right\rfloor \tag{1}
\]

in which, \( M \) is the available RAM, \( |V| \) is the total number of vertices in the graph, \( \varphi \) is the amount of data needed to store each vertex, and \( T \) is the number of threads. For example, for 1 GB RAM, a graph with 2 billion nodes, 2 threads, and 4 bytes of data per node, \( \beta = \left\lfloor \frac{2 \times 8 \times 2 \times 2 \times 10^9}{2 \times 10^9} \right\rfloor = 30 \), thus requiring \( 30^2 \times 900 \) blocks.

### 3.2 The M-Flash Processing Model

This section presents our proposed M-Flash. We first describe two novel strategies targeted at processing dense and sparse blocks. Next, we present the novel cost-based optimization strategy used by M-Flash to take the best of them.

#### Dense Block Processing (DBP):

Figure 2 illustrates the DBP processing; notice that vertex intervals are represented by vertical (Source I) and horizontal (Destination I) vectors. After partitioning the graph into blocks, we process them in a vertical zigzag order, as illustrated in Figure 2. There are three reasons for this order: (1) we store the computation results in the destination vertices; so, we can “pin” a destination interval (e.g., \( I^{(3)} \)) and process all the vertices that are sources to this destination interval (see the red vertical arrow); (2) using this order leads to fewer reads because the attributes of the destination vertices (horizontal vectors in the illustration) only need to be read once, regardless of the number of source intervals. (3) after reading all the blocks in a column, we take a “U turn” (see the orange arrow) to benefit from the fact that the data associated with the previously-read source interval is already in memory, so we can reuse that.

Within a block, besides loading the attributes of the source and destination intervals of vertices into RAM, the corresponding edges \( e = \langle \text{source, destination, edge properties} \rangle \) are sequentially read from disk, as explained in Fig. 3. These edges, then, are processed using a user-defined function so to achieve a given desired computation. After all blocks in a column are processed, the updated attributes of the destination vertices are written to disk.

**Streaming Partition Processing (SPP):** The performance of DBP decreases for graphs with very low density (sparse) blocks; this is because, for a given block, we have to read more data from the source intervals of vertices than from the very blocks of edges. For such situations, we designed the technique named **Streaming Partition Processing (SPP)**. SPP processes a given graph using partitions instead of blocks. A graph partition can be a set of blocks sharing the same source node interval – a line in the logical partitioning; or, similarly, a set of blocks sharing the same destination node interval – a column in the logical partitioning. Formally, a source-partition \( \text{SP}^{(p)} = \bigcup G^{(p,q)} \) contains all blocks with edges having source vertices in the interval \( I^{(p)} \); a destination-partition \( \text{DP}^{(p)} = \bigcup G^{(p,q)} \) contains all blocks with edges having destination vertices in the interval \( I^{(q)} \). For example, in Figure 1, \( \text{DP}^{(2)} \) is the union of the blocks \( G^{(1,2)}, G^{(2,2)} \) and \( G^{(3,2)} \).

For processing the graph using \( SPP \), we divide the graph in \( \beta \) source-partitions. Then, we process partitions using a two-steps approach (see Fig. 4). In the first step for each source-partition, we load vertex values of the interval \( I^{(p)} \); next, we read edges of the partition \( \text{SP}^{(p)} \) sequentially from disk, storing in a temporal buffer edges together with their in-vertex values until the buffer is full. Later, we shuffle the buffer in-place, grouping edges by destination-
Figure 4: Example I/O operations for step 1 of source-partition $SPP^1$. Edges of $SPP^1$ are combined with their source vertex values. Next, edges are divided by $\beta$ destination-partitions in memory; and finally, edges are written on disk. On Step 2, destination-partitions are processed sequentially. Example I/O operations for step 2 of destination-partition $DP^{(1)}$.

Finally, we store to disk edges in $\beta$ different files, one by destination-partition. After we process the $\beta$ source-partitions, we get $\beta$ destination-partitions containing edges with their source values. In the second step for each destination-partition, we initialize vertex values of interval $I^{(\beta)}$; next, we read edges sequentially, processing their values through a user-defined function. Finally, we store vertex values of interval $I^{(\beta)}$ on disk. The $SPP$ model is an improvement of the edge streaming approach used in X-Stream; different from former proposals, $SPP$ uses only one buffer to shuffle edges, reducing memory requirements.

Bimodal Block Processing (BBP): Schemes $DBP$ and $SPP$ improve the graph performance in opposite directions.

- how can we decide which processing scheme to use when we are given a graph block to process?

To answer this question, we propose to join $DBP$ and $SPP$ into a single scheme – the Bimodal Block Processing (BBP). The combined scheme uses the theoretical I/O cost model proposed by Aggarwal and Vitter [1] to decide for $SBP$ or $SPP$. In this model, I/O cost for an algorithm is equal to the number of blocks with size $B$ transferred between disk and memory plus the number of non-sequential seeks.

For processing a graph $G$, $DBP$ performs the following operations over disk: one read of the edges, $\beta$ reads of the vertices, and one writing of the updated vertices. Hence, the I/O cost for $DBP$ is given by:

$$\Theta(DBP(G)) = \Theta\left(\frac{(\beta + 1)|V| + |E|}{B} + \beta^2\right)$$

In turn, $SPP$ performs the following operations over disk: one read of the vertices and one read of the edges grouped by source-partition; next, it shuffles edges by destination-partition in memory, writing the new version $E$ on disk; finally, it reads the new edges from disk, calculating the new vertex values and writing them on disk. The I/O cost for $SPP$ is:

$$\Theta(SPP(G)) = \Theta\left(\frac{2|V| + |E| + |E|}{B} + \beta\right)$$

Equations 2 and 3 define the I/O cost for one processing iteration over the whole graph $G$. However, in order to decide in relation to blocks, we are interested in the costs of Equations 2 and 3 divided according to the number of blocks $\beta^2$. The result, after the appropriate algebra, reduces to Equations 4 and 5.

$$\Theta\left(DBP\left(G^{(p,q)}\right)\right) = \Theta\left(\frac{\vartheta \phi (1 + 1/\beta) + \xi \psi}{B}\right)$$

$$\Theta\left(SPP\left(G^{(p,q)}\right)\right) = \Theta\left(\frac{2\vartheta \phi/\beta + 2\xi \phi \psi + \xi \psi}{B}\right)$$

in which, $\xi$ is the number of edges in $G^{(p,q)}$, $\vartheta$ is the number of vertices in the interval, and $\phi$ and $\psi$ are, respectively, the number of bytes to represent a vertex and an edge $e$. Once we have the costs per block of $DBP$ and $SPP$, we can decide between one and the other by simply analyzing the ratio $SPP/DBP$:

$$\Theta\left(\frac{SPP}{DBP}\right) = \Theta\left(\frac{1}{\beta} + \frac{2\xi \phi}{\vartheta}\right)$$

This ratio leads to the final decision equation:

$$BlockType\left(G^{(p,q)}\right) = \begin{cases} \text{sparse,} & \Theta\left(\frac{SPP}{DBP}\right) < 1 \\ \text{dense,} & \text{otherwise} \end{cases}$$

We apply Equation 6 to select the best option according to Equation 7. With this scheme, $BBP$ is able to select the best processing scheme for each block of a given graph. In Section 4, we demonstrate that this procedure yields a performance superior than the current state-of-the-art frameworks.

3.3 Programming Model in M-Flash

M-Flash’s computational model, which we named $MAlgorithm$ (short for $Matrix\ Algorithm\ Interface$) is shown in Algorithm 1. Since $MAlgorithm$ is a vertex-centric model, it stores computation results in the destination vertices, allowing for a vast set of iterative graph computations, such as PageRank, Random Walk with Restarts (RWR), Weakly Connected Components (WCC), and diameter estimation.

The $MAlgorithm$ interface has four operations: initialize, gather, process, and apply. The initialize operation loads the initial value of each destination vertex; the gather operation collects data from neighboring vertices; the process operation processes the data gathered from the neighbors of a given vertex – the desired processing is defined here; finally, the apply operation stores the new computed values of the destination vertices to the hard disk, making them available for the next iteration. Note that initialize and apply operations are not mandatory, while process operation is used only in multithreaded executions.

To demonstrate the flexibility of $MAlgorithm$, we show in Algorithm 2 the pseudo code of how the PageRank algorithm...
Algorithm 2 PageRank in M-Flash

\[ \text{degree}(v) = \text{out degree for Vertex } v; \]
\[ \text{initialize (Vertex } v): \]
\[ v.\text{value} = 0; \]
\[ \text{gather (Vertex } u, \text{ Vertex } v, \text{ EdgeData data}): \]
\[ v.\text{value} + = u.\text{value}/\text{degree}(v); \]
\[ \text{process (Accum } v,1, \text{ Accum } v,2, \text{ Accum } v,\text{out}): \]
\[ v.\text{out} = v.1 + v.2; \]
\[ \text{apply (Vertex } v): \]
\[ v.\text{value} = 0.15 + 0.85 \times v.\text{value}; \]

(using power iteration) can be implemented. The input to PageRank is made of two vectors, one storing node degrees, and another one for storing intermediate PageRank values, initialized to 1/|V|. The algorithm’s output is a third node vector that stores the final computed PageRank values. For each iteration, M-Flash executes the \text{MAlgorithm} operations on the output vector as follows:

- **initialize:** the vertices’ values are set to 0;
- **gather:** accumulates the intermediate PageRank values of all in-neighbors u of vertex v;
- **process:** sums up intermediate PageRank values – M-Flash supports multiple threads, so the process operation combines the vertex status for threads running concurrently;
- **apply:** calculates the vertices’ new PageRank values (damping factor = 0.85, as recommended by Brian and Sergei [4]).

The input for the next iteration is the output from the current one. The algorithm runs until the PageRank values converge; it may also stop after executing one certain number of iterations defined by the user.

Many other graph algorithms can be decomposed into or take advantage of the same four operations and implemented in similar ways, including Weakly Connected Component (see Algorithm 4 in Appendix A), Sparse Matrix Vector Multiplication SpMV (Algorithm 5 in Appendix A), eigensolver (Algorithm 6 in Appendix A), diameter estimation, and random walk with restart [8].

3.4 System Design & Implementation

This section details the implementation of M-Flash. It starts processing the input graph stored in standard file formats, then, it transforms the graph to one flat array format in which each edge has a constant size. At the same time of graph preprocessing, M-Flash divides the edges in \( \beta \) source-partitions and it counts the number of edges by block. An edge \( e = (v_{\text{source}}, v_{\text{destination}}, \text{data}) \) belongs to block \( G^{(p,q)} \) when \( v_{\text{source}} \in I^{(p)} \) and \( v_{\text{destination}} \in I^{(q)} \). Blocks are classified in sparse or dense using Equation 7. Note that M-Flash does not sort edges by source or destination, it simply splits edges up to \( \beta^2 \) blocks, \( \beta^2 \leq |V| \). After all edges are preprocessed, whenever a source-partition contains dense blocks, M-Flash splits this partition between one sparse partition and dense blocks. The sparse partition contains all edges for the sparse blocks in the source-partition. The I/O cost for preprocessing is \( \frac{|E|}{\beta^2} \). Algorithm 3 shows the pseudocode of M-Flash. The aforementioned preprocessing refers to Step 4 of the algorithm. Sparse partitions are processed using SPP and dense blocks are processed using DBP. Algorithm 3 shows the overlap between models DBP and SPP, minimizing I/O cost and thus increasing performance.

4. EVALUATION

Overview: We compare M-Flash with multiple state-of-the-art approaches: GraphChi, TurboGraph, X-Stream, and MMap. For a fair comparison, we use experimental setups recommended by the authors of the other approaches in the best way that we can. We first describe the datasets used in our evaluation (Subsection 4.1) and our experimental setup (Subsection 4.2). Then, we compare the runtimes of all approaches for two well-known, essential graph algorithms (Subsections 4.3 and 4.4) that are available for all competing works. To demonstrate how M-Flash generalizes to more algorithms, we implemented the Lanczos algorithm (with selective orthogonalization) as an example, one of the most computationally efficient approaches to compute eigenvalues and eigenvectors [17, 8] (Subsection 4.5) — to the best of our knowledge, M-Flash provides the first design and implementation that can handle graphs with more than one billion nodes when the vertex data cannot fully fit in RAM (e.g., YahooWeb graph). Next, in Subsection 4.6, we show that M-Flash continues to run at high speed even when the machine has little RAM (including extreme cases, like when using solely 4GB), in contrast to other methods that slow down in such circumstance. Finally, through an
Table 1: Real graph datasets used in our experiments.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes</th>
<th>Edges</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiveJournal</td>
<td>4,847,571</td>
<td>68,993,773</td>
<td>Small</td>
</tr>
<tr>
<td>Twitter</td>
<td>41,652,230</td>
<td>1,468,365,182</td>
<td>Medium</td>
</tr>
<tr>
<td>YahooWeb</td>
<td>1,413,511,391</td>
<td>6,636,600,779</td>
<td>Large</td>
</tr>
</tbody>
</table>

4.1 Graph Datasets

We use three real graphs of different scales: a LiveJournal graph [2] with 69 million edges (small), a Twitter graph [11] with 1.47 billion edges (medium), and the YahooWeb graph [22] with 6.6 billion edges (large). Table 1 reports their numbers of nodes and edges.

4.2 Experimental Setup

All experiments were run on a standard desktop computer with an Intel i7-4770K quad-core CPU (3.50 GHz), 16 GB RAM and 1 TB Samsung 850 Evo SSD disk. Note that M-Flash does not require an SSD to run (neither do GraphChi and X-Stream), while TurboGraph does; thus, we used an SSD to make sure all methods can perform at their best. Table 2 shows preprocessing time for each graph using 8GB of RAM. As it can be seen, M-Flash has competitive preprocessing runtime, beating all the related works in most cases.

GraphChi, X-Stream and M-Flash were run on Linux Ubuntu 14.04 (x64). TurboGraph was run on Windows (x64) since it only supports Windows [6]. MMap was written in Java, thus we were able to run it on both Linux and Windows; we ran it on Windows, following MMap’s authors setup [13]. All the reported runtimes were given by the average time of three cold runs, that is, with all caches and buffers purged between runs to avoid any potential advantage gained due to caching or buffering effects.

The libraries are configured as follows:

- GraphChi: C++ version, downloaded from their GitHub repository in February, 2015. Buffer sizes configured to those recommended by their authors
- X-Stream: C++ v0.9. Buffer size desirably configured close to available RAM
- TurboGraph: V0.1 Enterprise Edition. Buffer size desirably configured to 75% of available RAM, the limit supported by TurboGraph, as observed by [13]
- MMap: Java version (64-Bit) with default parameters.
- M-Flash: C++ version. Freely available for download at https://github.com/M-Flash

We ran all the methods at their best configurations since we wanted to truly verify performance at the most competitive circumstances. As we show in the following sections, M-Flash exceeded the competing works both empirically and theoretically. At the end of the experiments, it became clear that the design of M-Flash considering the density of blocks of the graph granted the algorithm improved performance.

4.3 PageRank

Figure 5 shows how the PageRank runtime of all the methods compares.

- LiveJournal (small graph; Fig. 5a): Since the whole graph and all node vectors fully fit in RAM, all approaches finish in seconds. Still, M-Flash was the fastest, up to 3.3X of GraphChi, and 2.4X of X-Stream and TurboGraph.
- Twitter (medium graph; Fig. 5b): The edges of this graph do not fit in RAM (it requires 11.3GB) but its node vectors do. M-Flash had a similar performance, but for a few seconds, if compared to TurboGraph and MMap for two reasons: (a) the Twitter graph is less challenging as it has a homogeneous density, with all its blocks being sparse; (b) TurboGraph’s and MMap’s implementations were highly optimized as they do not provide a generic programming model, saving on function calls. In comparison with GraphChi and X-Stream i.e., the related works that offer generic programming models, M-Flash was fastest, at 1.7X to 3.6X speed.
- YahooWeb (large graph; Fig. 5c): For this billion-node graph, neither its edges nor its node vectors fit in RAM; this challenging situation is where M-Flash significantly outperforms the other methods. Figure 5(c) confirms this claim, showing that M-Flash is faster, at a speed that is 2.2X to 3.3X that of the other approaches.

4.4 Weakly Connected Component

When there is enough memory to store all the vertex data, the Union Find algorithm [19] is the best option to find all the Weakly Connected Components (WCC) in one single iteration. Otherwise, with memory limitations, an iterative algorithm produces identical solutions, as listed in Appendix A, Algorithm 4. Hence, in this round of experiments, we use Algorithm Union Find to solve WCC for the small and medium graphs, whose vertices fit in memory; and we use Algorithm 4 to solve WCC for the YahooWeb graph.

Figures 6(a) and 6(b) show the runtimes for the LiveJournal and Twitter graphs with 8GB RAM; all approaches use Union Find, except X-Stream. This is because of the way that X-Stream is implemented, which handles only iterative algorithms.

In the WCC problem, M-Flash is again the fastest method in respect to the entire experiment: for the LiveJournal graph, M-Flash ties with GraphChi, it is 9X faster than X-Stream, 7X than TurboGraph, and 1.5X than MMap. For the Twitter graph, M-Flash’s speed is only a few seconds behind GraphChi, 71X faster than X-Stream, 5X than TurboGraph, and 2.6X than MMap.

The results for the YahooWeb graph are shown in Figure 6(c), one can see that M-Flash was significantly faster than GraphChi, and X-Stream. Similarly to the PageR-
Figure 5: Runtime of PageRank for LiveJournal, Twitter and YahooWeb graphs with 8GB of RAM. M-Flash is 3X faster than GraphChi and TurboGraph. (a & b): for smaller graphs, such as Twitter, M-Flash is as fast as some existing approaches (e.g., MMap) and significantly faster than others (e.g., 4X of X-Stream). (c): M-Flash is significantly faster than all state-of-the-art approaches for YahooWeb: 3X of GraphChi and TurboGraph, 2.5X of X-Stream, 2.2X of MMap.

Figure 6: Runtimes of the Weakly Connected Component problem for LiveJournal, Twitter, and YahooWeb graphs with 8GB of RAM. (a & b): for the small (LiveJournal) and medium (Twitter) graphs, M-Flash is faster than, or as fast as, all the other approaches. (c) M-Flash is pronouncedly faster than state-of-the-art approaches for the large graph (YahooWeb): 9.2X of GraphChi and 5.8X of X-Stream.

4.5 Spectral Analysis using The Lanczos Algorithm

Eigenvalues and eigenvectors are at the heart of numerous algorithms, such as singular value decomposition (SVD) [3], spectral clustering [21], triangle counting [20], and tensor decomposition [10]. Hence, due to its importance, we demonstrate M-Flash over the Lanczos algorithm, a state-of-the-art method for eigen computation. We implemented it using method Selective Orthogonalization (LSO). Algorithm 6 shows a pseudocode implementation. This implementation demonstrates how M-Flash’s design can be easily extended to support spectral analysis of billion-scale graphs. To the best of our knowledge, M-Flash provides the first design and implementation that can handle Lanczos for graphs with more than one billion nodes when the vertex data cannot fully fit in RAM. M-Flash provides functions for basic vector operations using secondary memory. Therefore, for the YahooWeb graph, we are not able to compare it with the other competing frameworks using only 8GB of memory, as in the case of GraphChi.

To compute the top 20 eigenvectors and eigenvalues of the YahooWeb graph, one iteration of LSO over M-Flash takes 737s when using 8 GB of RAM. For a comparative panorama, to the best of our knowledge, the closest comparable result of this computation comes from the HEigen system [7], at 150s for one iteration; note however that, it was for a much smaller graph with 282 million edges (23X fewer edges), using a 70-machine Hadoop cluster, while our experiment with M-Flash used a single, commodity desktop computer and a much larger graph.

4.6 Effect of Memory Size

As the amount of available RAM strongly affects the computation speed in our context, we study here the effect of memory size.

Figure 7 summarizes how all approaches perform under 4GB, 8GB, and 16GB of RAM, when running one iteration of PageRank over the YahooWeb graph. M-Flash continues to run at the highest speed even when the machine has very little RAM, 4 GB in this case. Other methods tend to slow down. In special, MMap does not perform well due to thrashing, a situation when the machine spends a lot of time on mapping disk-resident data to RAM or un-mapping data from RAM, slowing down the overall computation. For 8 GB and 16 GB, respectively, M-Flash outperforms all the competitors for the most challenging graph, the YahooWeb. Notice that all the methods, but for our M-Flash and X-Stream, are strongly influenced by restrictions in memory size; according to our analyses, this is due to the higher number of data transfers needed by the other methods when not all the data fits in the memory. Despite that X-Stream
M-Flash was designed so that it is possible to integrate a wide range of popular graph algorithms according to its Matrix Algorithm Interface model, including the first single-machine billion-scale eigensolver. We perform these analyses by means of MathLab simulations that were validated empirically. We considered the characteristics of the three datasets used so far, LiveJournal, Twitter, and YahooWeb. For each case, we calculated the t-cost (y axis) as a function of the available memory (x axis), which, as we have seen, is the main constraint for graph processing frameworks.

Figure 9 shows that, for all the graphs, DBP-only processing is the least efficient when memory is reduced; however, when we combine DBP (for dense region processing) and SPP (for sparse region processing) into BBP, we benefit from the best of both worlds. The result corresponds to the best performance, as seen in the charts. Figure 10 shows the same simulated analysis of t-cost (y axis) in function of the available memory (x axis). But now with an extra variable: the density of hypothetical graphs, which is assumed to be uniform in each analysis. Each plot, from (a) to (d) considers a different density in terms of average vertex degree, respectively, 3, 5, 10, and 30. In each plot, there are two curves, one corresponding to DBP-only, and one for SSP-only; and, in dark blue, we depict the behavior of M-Flash according to the combination BBP. Notice that as the amount of memory increases, so does the performance of DBP (in light graph), which takes less and less time to process the whole graph (decreasing curve). SPP, in turn, has a steady performance, as it is not affected by the amount of memory (light blue line). In dark blue, one can see the performance of BBP; that is, which kind of processing will be chosen by Equation 7 at each circumstance. For sparse graphs, Figures 10(a) and 10(b), SSP answers for the greater amount of processing; while the opposite is observed in denser graphs, Figures 10(c) and 10(d), when DBP defines almost the entire dark blue line of the plot.

These results show that the graph processing must take into account the density of the graph at each moment (block) so to choose the best strategy. It also explains why M-Flash improves the state-of-the-art. It is important to note that no former algorithm considered the fact that most graphs present varying density of edges (dense regions with much more edges than other regions that are sparse). Ignoring this fact leads to decreased performance in the form of higher number of data transfers between memory and disk, as we empirically verified in the former sections.

5. CONCLUSIONS

We proposed M-Flash, a single-machine, billion-scale graph computation framework that uses a block partition model to maximize disk access speed. M-Flash uses an innovative design that takes into account the variable density of edges observed in the different blocks of a graph. Its design uses Dense Block Processing (DBP) when the block is dense, and Streaming Partition Processing (SPP) when the block is sparse; for taking advantage of both worlds, it uses the combination of DBP and SPP according scheme Bimodal Block Processing (BBP), which is able to analytically determine whether a block is dense or sparse and trigger the appropriate processing. To date, M-Flash is the first framework that considers a bimodal approach for I/O minimization.

We did not compare with TurboGraph because it runs only on Windows, which does not provide readily available tools for measuring I/O speed.

---

We considered the characteristics of the three datasets used so far, LiveJournal, Twitter, and YahooWeb. For each case, we calculated the t-cost (y axis) as a function of the available memory (x axis), which, as we have seen, is the main constraint for graph processing frameworks.
Figure 8: I/O operations for 1 iteration of PageRank over the YahooWeb graph. M-Flash performs significantly fewer reads and writes (in green) than other approaches. M-Flash achieves and sustains high-speed reading from disk (the “plateau” in top-right), while other methods do not.

Figure 9: I/O cost using DBP, SPP, and BBP for LiveJournal, Twitter and YahooWeb Graphs using different memory sizes. BBP model always performs fewer I/O operations on disk for all memory configurations.

Figure 10: I/O cost using DBP, SPP, and BBP for a graph with densities $k = \{3, 5, 10, 30\}$. Graph density is the average vertex degree, $\|E\| \approx k\|V\|$. DBP increases considerably I/O cost when RAM is reduced, SPP has a constant I/O cost and BBP chooses the best configuration considering graph density and available RAM.

experiments using large real graphs. M-Flash consistently and significantly outperformed all state-of-the-art approaches, including GraphChi, X-Stream, TurboGraph and MMap. M-Flash runs at high speed for graphs of all sizes, including the 6.6 billion edge YahooWeb graph, even when the size of memory is very limited (e.g., 6.4X faster than X-Stream, when using 4GB of RAM).

6. REFERENCES


APPENDIX

Algorithm 6 Lanczos Selective Orthogonalization

Input: Graph \( G(V,E) \)

Input: dense random vector \( b \) with size \( |V| \)

Input: maximum number of steps \( m \)

Input: error threshold \( \epsilon \)

Output: Top \( k \) eigenvalues \( \lambda[1..k] \), eigenvectors \( Y^{n \times k} \)

Using M-Flash’s provided functions for vector operations in secondary memory.

\[ \beta_0 \leftarrow 0, v_0 \leftarrow 0, v_1 \leftarrow b/\|b\| \]

for \( i = 1 \) to \( m \) do

\[ v \leftarrow Gv_i \]

\[ \alpha_i \leftarrow v_i^Tv \]

\[ v \leftarrow v - \beta_i v_{i-1} - \alpha_i v_0 \]

\[ \beta_i \leftarrow \|v\| \]

end for

Orthogonalization using two previous basis vectors:

\( T_i \leftarrow \{ \text{build tri-diagonal matrix from } \hat{\alpha} \text{ and } \hat{\beta} \} \)

Eigen decomposition:

\( QDQ^T \leftarrow EIG(T_i) \)

for \( j = 1 \) to \( i \) do

if \( \hat{\beta}Q[i,j] \leq \sqrt{\epsilon}||T_i|| \) then

end if

end for

if \( v \) was selectively orthogonalized) then

end if

end for

break loop

Compute eigenvectors. \( Q_k \) is the columns of \( Q \) corresponding to \( \lambda \)
Capítulo B. Processamento massivo de grafos
Fatoração matricial aplicada sobre a base de dados DBLP
Spectral analysis and text processing over the Computer Science literature: patterns and discoveries

Rosa V. E. Quille, Caetano Traina Jr., Jose F. Rodrigues Jr.
Instituto de Ciências Matemáticas e de Computação - USP
Avenida Trabalhador são-carlense, 400 - Centro
CEP: 13566-590 - São Carlos - SP
{encinas,caetano,junio}@icmc.usp.br

ABSTRACT
We defend the thesis that the use of text analytics can boost the results of analyses based on Singular Value Decomposition (SVD). To demonstrate our supposition, first we model the Digital Bibliography & Library Project (DBLP) as a relational schema; over this schema we use text analytics applied to the terms extracted from the titles of the articles. Then, we apply SVD on the relationships defined between these terms, publication vehicles, and authors; accordingly, we were able to identify the more representative communities and the more active authors relating them to the most meaningful terms and topics found in their respective publications. The results were semantically dense and concise, also leading to performance gains.

Categories and Subject Descriptors
G.1.3 [Numerical Analysis]: Numerical Linear Algebra-Singular Value Decomposition; H.2.8 [Database Applications]: Data mining

Keywords
DBLP, relational data, data analysis, matrix factorization, singular value decomposition

1. INTRODUCTION
We consider the Digital Bibliography & Library Project (DBLP) as a relational database over which we define an extensive analytical process based on text and linear-algebra analytical techniques. We model DBLP as a relational schema in which its entities (authors, events, vehicles, and terms) correspond to nodes of a graph representation, and its relationships correspond to edges. We use text analytic techniques to process the terms found on the articles' titles; then, we use Singular Value Decomposition (SVD), a powerful algebraic technique – also known as spectral decomposition – for matrix (graph) analysis.

2. RELATED WORK
Singular value decomposition (spectral analysis) has been explored in many scenarios. In the work of Prakash et al. [6], the authors obtain the SVD from Laplacian representations, and then plot the vectors of the first matrix of the decomposition against themselves in what they call EE-plots. They found that such plots are very informative in what concerns communities. In another work, Kang et al. [1] explain how to calculate SVD’s from billion-scale graphs, an operation that has several performance issues. To do so, they use MapReduce and Hadoop technologies for parallel distributed processing.

Leting et al. [7] perform spectral analysis over signed graphs. They found that such graphs have specific properties; in special, they found that when a graph is signed, its communities are more clearly observed, even if the connections among them tend to increase. Maruhashi and Faloutsos [5] introduce EigenDiagnostics, an algorithm that calculates and combines several spectral measures to spot patterns in graph-represented data. In another work, Kim et al. [2] apply spectral analysis over blog data (or blogosphere). They found a set of outstanding communities derived from the relationships drawn from blogs and posts; also, they were able to characterize and interpret the communities based on the key terms used to compose the posts.

Our work differs from former proposals as we combine text analytics with spectral analysis to gain deeper insight from the computer science literature – we aim at identifying main publication vehicles, authoritative authors, and prevalent communities, among other facts. In our experiments, we demonstrate that by refining the data before the SVD factorization, it is possible to pronouncedly improve the data analysis.

3. SINGULAR VALUE DECOMPOSITION
Technique SVD is a matrix factorization method widely used in applications such as signal processing and statistics [8]. Given an $A_{nxm}$ matrix, let $U$ be an nxr matrix whose columns are the singular vectors orthogonal to $\Lambda$, and $V^T$ be the rxm matrix whose columns are the singular vectors orthogonal to $A$; then $A$ can be defined by equality 1.

$$A_{nxm} = U_{nxr} \Lambda_{rx1} V_{rxm}^T = \sum_{i=1}^{\text{rank}(A)} (\lambda_i u_i \otimes v_i)$$

where the symbol $\otimes$ is the outer product of two given vectors, and $\text{rank}(A)$ is the rank of matrix $A$. The rank of a matrix is the number of linearly independent rows (or columns) in it; thus, $\text{rank}(A) \leq \min\{m,n\}$. 


We use SVD for two reasons: (1) Detect communities and outstanding elements, according to which, after the singular-factorization, the largest singular values will correspond to subsets (communities) in which the elements both of the rows and of the columns interact more intensely; also, the largest values found in the singular vectors of matrices $U_{n \times r}$ and $V^T_{r \times m}$ will correspond to line and column elements that are highly active, spotting outstanding elements for the sake of analysis; and (2) Dimensionality Reduction. DBLP data has a high dimensionality in number of rows and columns; however, its intrinsic dimensionality is not that high, as many authors are inactive. It means that many dimensions are redundant in respect to the most significant data; for this reason we use technique low-rank approximation in order to restrict data to its most significant elements. This technique is given by:

$$A_{n \times m} \approx U_{n \times r} \Lambda_{r \times r} V^T_{r \times m}$$

(2)

where $\Lambda$ contains only the largest singular values, with the others being replaced by zero.

4. METHODOLOGY

Here we present an overview of the seven steps of our methodology: (1) Pre-processing and cleaning, (2) Modeling, (3) Relational transformation, (4) Selection, (5) Processing SVD, (6) Analysis - SVD, and (7) Interpretation/Evaluation – as illustrated in Figure 1. In the following sections, we provide details of each step.

4.1 Pre-processing and cleaning

DBLP data is full of redundancy, lack of conformity, non-homogeneity, and noise. Therefore, before we can start processing it, we must clean it up using multiple techniques applied to the papers' titles, names of authors, events, and vehicles. Our cleaning step includes the following techniques: (a) ASCII conversion of characters; (b) tokenization, the first step for text preparation in areas as natural language processing (NLP) and information retrieval (IR); (c) removal of stopwords, words that appear with high frequency in text sentences, but that have no content that can help in interpretation; and (d) stemming of the terms of the papers' titles, the combination of different forms of a word in a word representative joint, the stem.

4.2 Data modeling and relational transformation

Since DBLP is available in XML semi-structured format, it is not readily adequate to be represented as a graph; XML demands intense parsing operations, and does not support aggregation for the task of weighting the edges of the graph representation. Therefore, we firstly described DBLP as an entity-relationship model, further transforming it into a relational database. The model contains many-to-many relationships that describe the same information as that comprised by weighted graphs. This property is the focus of our analysis by means of SVD algebra.

4.3 Data selection

As presented in Table 1, the main entities of the database are Author and Article; the essence of DBLP's literature orbit is this two entities. The former corresponds to 1,054,199 instances and the latter corresponds to 1,801,576 instances. These numbers are by far too large for algebraic processing, therefore we filtered them out by analyzing their distribution.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Number of entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>1,054,199</td>
</tr>
<tr>
<td>Articles</td>
<td>1,801,576</td>
</tr>
<tr>
<td>Events</td>
<td>3,050</td>
</tr>
<tr>
<td>Vehicles</td>
<td>4,262 (1,137 journals and 3,125 conferences)</td>
</tr>
</tbody>
</table>

### 4.3.1 Authors selection

One thing about DBLP is that it is heavily unbalanced in respect to its authors' production in number of articles. To verify this aspect closely, we plotted the Authors-Articles distribution in Figure 2(a). The plot shows a long-tail distribution in which the majority of authors has no more than 22 articles - more precisely this portion corresponds to 1,016,354 authors, or ~ 96% of the instances. DBLP has also over 80 authors with more than 300 articles, and a champion author with over 600 articles. The plot clearly depicts a power-law distribution according to which the number of authors ($y$) having a certain number of articles ($x$) varies as a power of the number of articles, that is:

$$y \propto x^{-2.06}$$

(3)

If we look close, this distribution is an instance of the Pareto Principle, or 80-20 rule, as ~ 24% of the authors are responsible for ~ 76% of the articles; while ~ 76% of the authors are responsible for only ~ 24% of the articles – see Figure 2(b). This leaded us to restrict our analysis to the 24% more prolific authors, that is, the 255,455 authors with 4 or more articles.

![Figure 2: Authors-Articles distribution. (a) Number of articles x Number of authors. (b) Rank-plot on Author index x Number of articles.](image-url)

### 4.3.2 Terms selection

Articles carry more information than simply defining instances; all of them have a title composed of a set of semantic terms. The semantics of such terms can be used to interpret and correlate DBLP data with richer details because specific terms address specific areas and research interests. By considering the terms of the titles instead of the articles, we got two advantages: since many terms appear recurrently in the titles, there are 292,919 terms, much less than the number of articles; and, interpreting key terms is simpler than interpreting complex titles. The drawback is that many terms appear in a frequency high enough to prevent valuable interpretation (e.g., “complex” and “efficient”); therefore, in this
In order to choose the terms, we calculated their term frequency (TF); thus, we counted them and plotted the results. In Figure 3, we can see the most frequent terms x the term rank. There we can observe two things: (1) there are too many terms with very low frequency; these terms are considered irrelevant to semantic text analysis; (2) there are many terms with very high frequencies - these terms are also considered irrelevant.

Hence, to restrict our space of terms, we interpreted the distribution of the terms according to Lunh’s work [4] and Zipf’s Law, which states that the most significative terms (TS) are those that are not too common, and neither those that are too rare; and, also, that the significance of the terms is given by the Normal Distribution centered on the mean (µ) of the TF-rank and standard deviation (σ), and given by equation 4:

$$TS(i) = f(r_i; \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\left(r_i - \mu\right)^2 / 2\sigma^2}$$  (4)

where $r_i$ is the TF-rank of the term $i$, and $TS(i)$ is the significance value of the term $i$.

In our case $\mu = 2033.6$ and $\sigma = 412.1392$, what gives us the plot seen in Figure 4(a). In the plot, the terms near the left end are high frequency terms, which are generally too common to be significant; the terms near the far right are low frequency terms, too rare to be significant. Therefore, the most useful terms are in the mid-range, the core of the importance given by a Normal Distribution. We use the techniques of Liu and Hoeber [3] in order to algorithmically identify the best cut; the technique generated the curve seen in Figure 4(b), which corresponds to re-calculated importance of the terms after processing. Finally, we were left with the 4,061 most significant terms.

5. EXPERIMENTS: SINGULAR VALUE DECOMPOSITION AND ANALYSIS

For our experiments, we use datasets of vehicles (all the conferences, workshops, and journals), authors, and terms, selected as described in subsections 4.3.1 and 4.3.2. With these sets of entities, we considered two experimental cases as summarized in Table 2.
Table 2: Dataset configurations used in the experiments.

<table>
<thead>
<tr>
<th>case 1: Terms (Luhn) x Vehicles (Luhn)</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terms</td>
<td>4,061</td>
</tr>
<tr>
<td>Vehicles</td>
<td>3,014</td>
</tr>
</tbody>
</table>

Table 3: Terms (Luhn) x Vehicles (Luhn) – top six frequent terms in six communities.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Most frequent terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2: image processing and computer vision</td>
<td>imag, video, segment, recognit, detect</td>
</tr>
<tr>
<td>C5: software engineering and web</td>
<td>data, inform, softwar, web, manag</td>
</tr>
<tr>
<td>C9: bio and parallel computing</td>
<td>simul, protein, data, gene, parallel</td>
</tr>
<tr>
<td>C16: interaction and multimedia</td>
<td>fuzzi, interact, design, comput, video</td>
</tr>
<tr>
<td>C20: control systems and programming</td>
<td>control, comput, stabil, linear, program</td>
</tr>
<tr>
<td>C29: chemical data processing</td>
<td>molecular, chemic, structur, studi, calcul</td>
</tr>
</tbody>
</table>

Table 4: Terms (Luhn) x Vehicles (Luhn) – top six conference and journals in the second and sixteenth community.

<table>
<thead>
<tr>
<th>Vehicles (C2)</th>
<th>Score</th>
<th>#Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf/icip</td>
<td>0.5670</td>
<td>13427</td>
</tr>
<tr>
<td>journals/ieicet</td>
<td>0.1353</td>
<td>12106</td>
</tr>
<tr>
<td>conf/icra</td>
<td>0.1174</td>
<td>11694</td>
</tr>
<tr>
<td>conf/iccc</td>
<td>0.1087</td>
<td>6439</td>
</tr>
<tr>
<td>conf/vtec</td>
<td>0.1427</td>
<td>6875</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vehicles (C16)</th>
<th>Score</th>
<th>#Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf/chi</td>
<td>0.2497</td>
<td>6903</td>
</tr>
<tr>
<td>conf/hci</td>
<td>0.2580</td>
<td>6323</td>
</tr>
<tr>
<td>journals/iss</td>
<td>0.2017</td>
<td>2669</td>
</tr>
<tr>
<td>conf/fuzzi</td>
<td>0.1804</td>
<td>6323</td>
</tr>
<tr>
<td>journals/nar</td>
<td>0.1394</td>
<td>2419</td>
</tr>
</tbody>
</table>

In this second experiment, we could observe that the SVD processing was significantly improved by the selection of terms based on the semantic filtering. In this case, the communities were better characterized both in terms and in vehicles, providing an insightful panorama of the computer science research.

5.2 Case 2: Terms (Luhn) x Authors

In the last experiment, we joined authors and the terms of the titles of the papers published by these authors. The analysis, here, is supposed to indicate communities according to the collaboration in between researchers rather than the topics of vehicles. The SVD decomposition indicated a large number – over 200 – of significant singular values in \( \lambda_r \), as illustrated in Figure 6, an expected behavior since the communities of authors obey to research groups, geographical proximity, and affinity in the area of expertise. As a sample, we present two highly significant terms for 6 major communities in Table 5. Based on these terms, we were able to characterize each community and, also, to estimate how active each community is, what is expressed by the order in the table.

In Table 6, we used the score of matrix \( V_{\times m} \) in order to track the most active authors in communities C2 and C4; along with the score, we validated the profile of the authors by checking the number of citations of each one at Microsoft Academic Research repository. For example, in community C2 the most important author is Thomas S. Huang with...
19,988 citations; he is in the top 6 of the Most Prolific DBLP Authors, having Pattern Recognition and Computer Vision as his primary research area. The second most important author is Wen Gao, with 16,336 citations, who is the top 4 most prolific author of DBLP; his research interests are Pattern Recognition, Computer Vision and Multimedia. In the same way, in community C4 the most authoritative author is Sudhakar M. Reddy, with 8,119 citations, who is the 573rd most prolific author of DBLP; his research interests are Distributed and Parallel Computing. The second author is Irith Pomeranz, with 5,563 citations; top 506 in DBLP; and Distributed and Parallel Computing as his research interests. These data demonstrated that the SVD method can be quite effective and interpretable. Moreover, this analysis is also applicable to other data domains with precision and interesting analytical properties.

Figure 6: Scree-plot of the energy levels of the SVD for Terms (Luhn) x Authors.

Table 5: Terms (Luhn) x Authors – most frequent terms in six communities.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Most frequent terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2: Pattern recognition</td>
<td>imag, recogn, video,</td>
</tr>
<tr>
<td>and image retrieval</td>
<td>learn, segment</td>
</tr>
<tr>
<td>C4: Graph Theory</td>
<td>graph, problem, algo-</td>
</tr>
<tr>
<td></td>
<td>rhythm, program, log-</td>
</tr>
<tr>
<td>C12: Image and Video</td>
<td>web, servic, video,</td>
</tr>
<tr>
<td>Processing</td>
<td>fuzzi, problem</td>
</tr>
<tr>
<td>C28: Intelligent</td>
<td>databas, mobil, fuzzi,</td>
</tr>
<tr>
<td>Information and</td>
<td>queri, perform</td>
</tr>
<tr>
<td>Database Systems</td>
<td></td>
</tr>
<tr>
<td>C39: Visualization</td>
<td>optim, visual, mobil,</td>
</tr>
<tr>
<td></td>
<td>semant, distribut</td>
</tr>
<tr>
<td>C50: Bioinformatics</td>
<td>struct, protein, pre-</td>
</tr>
<tr>
<td></td>
<td>dict, scheme, sequenc</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

We presented an extensive analytical process suitable for the relationships observed in relational databases. The proposed process departs from the representation of the references in between tuples (vertices) of the database as edges of a graph, and involves techniques of text analytics combined with Singular Value Decomposition. This way, our methodology is able to identify communities and outstanding elements according to the interrelationship that the elements of the database define in the context of the relational schema. We performed experiments over the DBLP dataset in order to defend the thesis that the use of text analytics can boost the results of SVD analysis. Our results demonstrated that, with the aid of text-based techniques, the output of the

Table 6: Terms (Luhn) x Authors – top six authoritative authors in the second and fourth communities.

<table>
<thead>
<tr>
<th>Author (C2)</th>
<th>Score</th>
<th>#Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thomas S. Huang</td>
<td>0.1115</td>
<td>605</td>
</tr>
<tr>
<td>Wen Gao</td>
<td>0.0840</td>
<td>606</td>
</tr>
<tr>
<td>HongJiang Zhang</td>
<td>0.0645</td>
<td>295</td>
</tr>
<tr>
<td>Chin-Chen Chang</td>
<td>0.0582</td>
<td>645</td>
</tr>
<tr>
<td>Edwin R. Hancock</td>
<td>0.0576</td>
<td>536</td>
</tr>
<tr>
<td>Barry L. Nelson</td>
<td>0.0552</td>
<td>255</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Author (C4)</th>
<th>Score</th>
<th>#Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sudhakar M. Reddy</td>
<td>0.0652</td>
<td>525</td>
</tr>
<tr>
<td>Irith Pomeranz</td>
<td>0.0636</td>
<td>448</td>
</tr>
<tr>
<td>Noga Alon</td>
<td>0.0525</td>
<td>443</td>
</tr>
<tr>
<td>Marek Karpinski</td>
<td>0.0467</td>
<td>268</td>
</tr>
<tr>
<td>Alan M. Frieze</td>
<td>0.04637</td>
<td>270</td>
</tr>
<tr>
<td>Wil M. P. van der Aalst</td>
<td>0.0424</td>
<td>280</td>
</tr>
</tbody>
</table>

SVD process is pronouncedly denser in terms of semantic and specificity. Our results are explained by the use of techniques that rely on Luhn’s theory and on Zipf’s law, what, for the DBLP domain, produced a semantically concentrated core of data to be processed before the SVD analysis; as a consequence, we achieved significant performance gains as well.

Acknowledgements

This work received support from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq-560104/2010-3), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP-2011/13724-1) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes).

7. REFERENCES

Apêndice D

Técnicas de análise de grafos aplicadas sobre a DBLP
Multimodal graph-based analysis over the DBLP repository: critical discoveries and hypotheses

Gabriel P Gimenes, Hugo Gualdron, Jose F Rodrigues Jr
University of Sao Paulo
Av Trab Sao-carlense, 400
Sao Carlos, SP, Brazil - 13566-590
{ggimenes,gualdron,junio}@icmc.usp.br

Mario Gazziero
Fed. University of Santo Andre
Av dos Estados, 5001
Santo Andre-SP-Brazil - 09210-580
mario.gazziero@ufabc.edu.br

1. INTRODUCTION

The properties and evolution of real-world networks are relevant topics nowadays, when network and mobile technologies are matured and disseminated. Network-like data arise, along the time dimension, from multiple domains in the order of hundreds of thousands of entities (nodes) and millions of relationships (edges). In respect to this matter, we study such networks by characterizing a particular type of data, that of the Computer Science literature - a network of co-authoring, co-editing of publications, and co-publication in periodicals. Scientific collaboration answers for a broad scope of interest; not only authors and editors, but the funding agencies and the society demand knowledge about how scientists behave concerning their collective production. As so, the analysis of Digital Bibliography & Library Project (DBLP) 1, one of the world’s major Computer Science literature repositories, from a network perspective can bring insights about the academic field and its future development.

Discovering non-evident facts about DBLP is not a trivial task, therefore we introduce what we call multimodal analysis, an ensemble of analytical techniques each with a different characteristic. We rely on statistical (degree, and weakly-connected components distribution), topological (average clustering coefficient, and effective diameter evolution), algorithmic (link prediction/machine learning), and algebraic techniques to inspect non-evident features of DBLP at the same time that we interpret the heterogeneous discoveries found along the work. As a result, we have put together a set of techniques demonstrating over DBLP what we call multimodal analysis, an innovative process of information understanding that demands a wide technical knowledge and a deep understanding of the data domain. We expect that our methodology and our findings will foster other multimodal analyses and also that they will bring light over the Computer Science research.

Categories and Subject Descriptors
G.2.2 [Discrete Mathematics]: [Digital Libraries] Graph Theory

General Terms
Network analysis, graph analysis, DBLP

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SAC 20/5 April 13 - 17 2015, Salamanca, Spain
ACM Copyright 2015 ACM 978-1-4503-3196-8/15/04 $15.00.
http://dx.doi.org/10.1145/2695664.2695699

1http://www.informatik.uni-trier.de/ ley/db/
perspective of the practices of the Computer Science community. Moreover, we present our methodology as a generic model that for the analysis of similar networks - our work comes as a systematic analytical process to be reproduced by academic peers and by practitioners.

2. RELATED WORK

There are plenty of studies that use Social Network Analysis (SNA) to transform network data into knowledge. Osiek et al. [17] try to answer whether attending conferences tend to increase scientific collaboration. To do so, the authors assume that having papers in the same conference correspond to a chance of conference-induced collaboration. With simple counting, they drew their conclusions by tracking the first common conference of each pair of authors and the first paper they wrote together. Conclusively, only 4.61% of the pairs of authors satisfied their supposition. Z. Huang et al. [9] used the Clique Percolation Method [15] to monitor the presence and the size of semantic communities over DBLP; they identified giant and small communities, each one with peculiarities about content, size and evolution. According to the notion of centrality, Leydesdorff [14] uses measures degree, betweenness, and closeness to evaluate the interdisciplinarity that is found in journals - although inconclusive, the author brings light to the problem.

Bollen et al. [3] collected online requests for electronic publications (clickstream) from Thomson Scientific, Elsevier, JSTOR, Ingenta, University of Texas, and California State University. Then, with metrics PageRank and betweenness they built a science map based on data from a broader audience updated in real time. J. Huang et al. [8] investigates a fragment of the Computer Science CiteSeer Digital Library; the authors performed a three-level analysis: network level, community level, and individual level. Their conclusions compare Database and Artificial Intelligence communities and introduce a Stochastic Poisson model to predict future collaboration behavior.

These previous works aim at characterizing the properties of nodes alone or, at last, the global properties of the structure by means of single metrics. In this work, we analyze the DBLP data by drawing the statistical distribution of several of its properties, and by drawing metrics that consider the time dimension.

In a different line, Leskovec et al. [11] present an extensive work on collecting metrics from a time evolving graph. In this work, the authors discuss the dynamics of viral marketing based on a large set of metrics, and on a recommendation propagation model. More recently, Benevenuto et al. [2] collected and analyzed time-evolving clickstream data from a large social network and, through statistical measures, deduced many aspects of its behavior. Finally, Hufsker et al. [10] describe an interesting analysis of interaction patterns on a virtual world environment; they do so by means of multiple measures such as shortest path, group similarity, clustering coefficient, and largest connected component. In conclusion, the authors describe the role of collective structures in determining the conduct of its members.

Recently, Aiello et al. [1] described how friends that have similar profiles (homophily) tend to get interconnected. In their study, the authors consider the groups to which the users belong, and the annotations (tags) of the users, among other features. With these features, the authors calculate the similarity between users, proposing a similarity threshold to state whether two users are to define a connection, or not. Regardless of its significative results, this study extrapolates the topological information of the network; it relies on information that, often, is not available or is not well-defined. This same limitation is faced by Brandao et al. [4] and Lim et al. [16].

By considering static and dynamic analytical approaches, as those presented in this section, we propose a multi-faceted analysis of DBLP – section 3.1. In our work, we draw conclusions from different points of view, static and dynamic, and from diverse complementary metrics. As so, we introduce necessary concepts in section 3, describe their application in section 4, and draw our conclusions in section 5.

3. MATERIAL AND METHODS

3.1 Digital Bibliography & Library Project (DBLP)

We used DBLP, one of the largest Computer Science bibliographic repositories available and, now, part of the ACM SIGMOD Anthology project. DBLP includes journals and conference proceedings since year 1936 - the main fields of its data entries are title, publication, authors, year of publication, page numbers, and editors – among others. In our investigation, we used records dated between 1970 and 2011 with the goal of characterizing the DBLP research community globally and along its evolution. Table 1 lists the cardinality of each entity extracted from DBLP used in our analysis.

Table 1: Number of entities involved in our analysis.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>1,069,321</td>
</tr>
<tr>
<td>Articles</td>
<td>1,801,376</td>
</tr>
<tr>
<td>Events</td>
<td>4,634</td>
</tr>
<tr>
<td>Publications</td>
<td>4,262</td>
</tr>
</tbody>
</table>

DBLP is available as an XML file that demands specific software for parsing its semi-structured data. It is a reference collection whose data quality is an important worry of its custodians; nevertheless, it presents some minor problems like name ambiguity and lack of data standardization. These problems prevented us from thoroughly using its content; not, however, causing prejudice to our analysis, as less than 5% of the data could not be used.

In order to use DBLP benefiting from the facilities of a Database Management Systems, we parsed it migrating its data to a rigid relational structure, from what we derived relationships. Figure 1 shows the data modeling, which has four many-to-many relationships: author-authorship-article, author-publishes-in-publication, author-attends-conference, author-contributed-to-conference.

---

2http://citeseer.ist.psu.edu

3http://www.sigmod.org/publications/sigmod-record/anthology

1130
and author-edits-conference; another point is that conference is a weak entity of publication, what means that every conference must have a correspondent publication (journal, proceedings, or book).

Figure 1: Entity-relationship modeling of our data.

Defining co-relationships from DBPL
From the original data of DBLP, we are interested in analyzing author co-relationships as, for example, given many-to-many relationship “Author (A) publishes (R) Article (B)”, the object of our study is the intrinsic co-authoring relationship. As a formal example, the “publishes” relationship can be represented in relational notation as $A = \{pk_a, name\}, B = \{pk_b, title\}$ and $R = \{pk_a, pk_b\}$. In this case, we are interested in the authors that published papers together; that is, we need a relation given, in relational algebra, by co-authorship $\leftrightarrow \Pi(A.pk_a, R.pk_b)(\sigma(A.pk_a = R.pk_b)(A \bowtie R.pk_a \neq R.pk_b \land R.pk_b \neq R.pk_a \land R.pk_a \neq R.pk_b \land \rho_R(R')))$. The result is relation co-authorship $= \{author, author', count\}$ used in our analysis. Furthermore, we created relations co-participation for authors who had papers at the same conferences, co-publication for those who had papers in the same journal, and co-edition for those who appear as editors of the same event or journal. Table 2 summarizes the datasets:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-authorship</td>
<td>Authors who published papers together.</td>
</tr>
<tr>
<td>Co-participation</td>
<td>Authors who had papers in the same conference.</td>
</tr>
<tr>
<td>Co-publication</td>
<td>Authors who had papers in the same journal.</td>
</tr>
<tr>
<td>Co-edition</td>
<td>Authors who appeared as editors of the same event or journal.</td>
</tr>
</tbody>
</table>

We created these relations in two versions: one having the number of times the relationship occurred, the other including the year when the relationship first took place. Respectively, they correspond to static weighted graphs and to graphs that evolve along time, both undirected for our experiments.

3.2 Methods
We make use of a number of social network metrics and techniques to inspect the characteristics of DBLP in complementary fashion. We apply these techniques either for the entire static network, or for consecutive annual snapshots of it. We present the results as summarizing plots and tables along the spectrum of each variable. We make use of the following methods:

- weakly-connected components (WCC) distribution: a WCC is an undirected subgraph in which every node has a path to every other node; counting the sizes of the connected components indicates how integrated are the research sub communities of the network;
- average clustering coefficient (ACC): global tendency of nodes to form clusters, or communities within the network – in a network of authors, it refers to the property of transitivity (presence of triangles), or, how likely the co-authors of my co-authors will become my co-authors; the coefficient ranges from 0 to 1, higher values indicating higher tendency of clustering;
- degree distribution (densification): the counting of the number of nodes with each given degree answers for how intense authors interact one with each other revealing important aspects of the evolution of the network;
- effective diameter evolution: the length of the 90th percentile path between any pair of nodes considered over time – states whether it happens and how intense is the small world phenomenon;
- predictability: refers to algorithmically calculating the probability that given existent vertices of a network will define new associations; it is based on the assumption that the past and the present behavior of the net can indicate what may happen in the future.

We apply these measures considering the relationships listed in Table 2.

4. MULTIMODAL ANALYSIS OF DBLP
We coin the term multimodal analysis, in the data mining context, referring to it as any ensemble of techniques from at least three distinct categories of data analysis. In this work, we instantiate this modality of knowledge discovery considering statistical, topological, algorithmic, and algebraic techniques; as explained in the following sections.

4.1 Weakly-connected components distribution (WCC)
Initially, we have verified interesting facts about WCC distribution over DBLP. Figure 2 depicts the case for co-authorship; in the figure we see that only 13% of the nodes form small components with up to 30 nodes; meanwhile, a giant component formed by nearly 87% of the nodes (~10^6 authors) defines a huge network in which researchers of the entire world share scientific expertise. Also interesting, the smaller components account for over 44,000 co-authorship sub-networks (~120,000 authors); a fact not so easy to
explain, but that, probably, corresponds to the eventual researchers that got involved with the academy only for the sake of obtaining a degree, without further research activity. Another possible explanation, comes from the white papers divulged by the industry; these papers aim at divulging new techniques or processes without a strict scientific contextualization.

Another possible explanation, comes from the white papers divulged by the industry; these papers aim at divulging new techniques or processes without a strict scientific contextualization.

4.2 Average Clustering Coefficient (ACC)

An expected property of co-authoring graphs is the presence of significant values of ACC. Through the co-authorship Node degree × ACC plot – Figure 3, we verified that this property is prominent for DBLP. High values (close to 1) of ACC are observed only for nodes with degree up to around 10, with values decreasing along the sequence so that the following power law is observed: \( \text{ACC} \propto \text{degree}^{-1.06} \). It means that nodes with degree up to 10 tend to have their connections highly interconnected, hence, they all together tend to form clusters (sets of nodes highly interconnected). This tendency decreases along with the increase of the degree.

4.3 Densification

From the degree-distribution plot – Figure 4, one can see that, along time, DBLP’s degree distribution is obeying to a power law with exponent \( \gamma = -1.36 \) (approximately). According to Leskovec et al. [12], this fact indicates that as more nodes enter DBLP, more edges appear following to the exponential relation:

\[
e(t) \propto n(t)^a
\]

where \( e(t) \) is the number of edges and \( n(t) \) is the number of nodes, and \( a \) is a specific exponent dictated by the slope \( \gamma = -1.36 \) of the degree distribution.

More specifically, according to Leskovec, \( a = 2/\gamma \) for the case in which \( 1 \leq \gamma \leq 2 \), or \( a = 2/1.36 = 1.47 \) for DBLP. This process is called densification, which answers for the intensity of according to which new edges appear in the network.

One can observe similar power law densification in other environments, as the web, where links correspond to new edges. However, one might wonder why the number of edges grows exponentially in an environment where new edges are not as cheap as in the web; but that, rather, depend on the publication of lengthy elaborated papers. We presume that two facts help to explain this tendency. First, master and Ph.D. titles were, originally, certifications of knowledge and experience - granted on an ad hoc basis; in the last decades, though, they became regular courses with well-defined time schedules and expected production. Consequently, a demand for “where to publish”, rather than “what to publish”, was created. This fact has led to a scientific literature that is prolix and that presents varying degrees of quality – more does not necessarily means better. Second, private and public science-funding agencies have demanded results in the form of publications as a condition for keeping up with their financing. Hence, researchers are pressed for numbers, be it good or bad – a straight consequence of this fact is the increasing number of authors per paper; in some cases, the so-called “academic collaboration” does not always corresponds to intellectual guidance and labor, but to co-financing and personal exchange as a means to increase one’s production.

DBLP obeys to a stable power-law distribution. Although the network is huge and mostly connected, new nodes cannot alter the main properties of the entire structure. One might think that a massive introduction of new nodes and edges could do so; however, as big as it can be, new generations of nodes and edges are still small if compared to the mega structure of DBLP. The equilibrium presented by DBLP is
observed in other systems as well [13] – as the respiratory system, automobile networks, and other social networks, being an instance of a well-defined natural phenomenon. As observed in the seminal work of Faloutsos [6], the exponent of its correspondent power-law distribution personifies this equilibrium.

### 4.4 Diameter

By inspecting the effective diameter evolution of the co-edition network, Figure 5, it is possible to see that the effective diameter starts to shrink after a certain point in time – around the year 1995. This fact suggests that, before this time there were new publication vehicles – with new editors appearing as well – showing up in the research community until a pick, when the same editors started to edit/co-edit for existing vehicles. It also suggests that after then, the number of new edges entering the network was much higher than the number of new nodes, what initiated a densification period. A possible explanation is that the committees of editors tend to have the same members that alternate between a limited set of possible committees, year after year. As so, the distance in between any two editors tends to decrease along the time. Possibly, this is the case because editing publications is a task that demands higher experience and expertise, characteristics of a limited set of researchers; moreover, as the number of editors is quite limited, the activity of edition is a matter of dispute in the community, what poses additional obstacles for newcomers.

### 4.5 Co-authoring predictability

Also relevant is to evaluate how predictable DBLP is. We do this by means of link recommendation techniques together with measures of accuracy. Here we verify this property by extracting metrics from the co-authoring network of DBLP; specifically, for each author we extract Number of common authors (a), Jaccard’s coefficient (b), Preferential attachment (c), Adamic-Adar coefficient (d), Resource allocation index (f), and Local path (g) – please check the work of Gimenes et al. [7] for details. These metrics are then used with supervised machine learning classifiers [19] J48, Naïve Bayes, Multilayer Perceptron, Bagging, and Random Forest, all of them available in the Weka framework, developed by the University of Waikato [5]. That is, we give some existing (past) co-authorings from 1995 to 2005 to the algorithms and they are expected to predict (‘future’) new co-authorings from 2006 to 2007 – of course, the algorithms are not given the answer, we use it to measure accuracy based on 10-fold cross-validation.

We performed link prediction for different profiles of authors as indicated by their degree; we considered authors with at least $d \geq 1$ existing co-authorings, with at least $d \geq 2$ existing co-authorings, and so on until at least $d \geq 8$ existing co-authorings. This extra parameter was set as a way to compare the predictability of the less ($d \geq 1$) and of the more ($d \geq 8$) active authors – pondering whether their profile is a relevant factor.

The accuracy of this methodology indicates how predictable DBLP is. We use measures Precision, Recall, F-Measure, and Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC). The higher the values the more predictable DBLP is because the metrics will indicate that we were successfully able to foresee new co-authorings. Table 3 presents the results for each minimum degree $d$ (line sets) and for each classifier (line) across each metric (columns). In the table we can see the highest values in bold; the best values (Random Forest classifier) are all above 0.8 (maximum is 1.0 – 100%), ranging from 0.867 to 0.977. Despite some classifiers having performed not well (NB and MLP), the others indicate a quite reasonable predictability, especially for minimum degrees above 1.

The predictability of DBLP indicates a network in which researchers do not interact intensely with fields not related to their own (further in the graph), what translates to low multidisciplinarity; it also indicates an interaction pattern in which authors tend to collaborate with the same authors recurrently or with co-authors of their co-authors (in triangle fashion). The fact that less active authors ($d \geq 1$) are harder to predict possibly corresponds to casual researchers that abandon the academy after getting their degree, and that do not aim at interacting with no other authors at all.
<table>
<thead>
<tr>
<th>d</th>
<th>Classifier</th>
<th>PREC</th>
<th>REC</th>
<th>F-MEAS</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>J48</td>
<td>0.723</td>
<td>0.706</td>
<td>0.7</td>
<td>0.764</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.741</td>
<td>0.585</td>
<td>0.505</td>
<td>0.626</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.562</td>
<td>0.555</td>
<td>0.541</td>
<td>0.593</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.809</td>
<td>0.8</td>
<td>0.798</td>
<td>0.887</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.877</td>
<td>0.868</td>
<td>0.867</td>
<td>0.939</td>
</tr>
<tr>
<td>2</td>
<td>J48</td>
<td>0.787</td>
<td>0.759</td>
<td>0.753</td>
<td>0.817</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.777</td>
<td>0.598</td>
<td>0.52</td>
<td>0.648</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.628</td>
<td>0.618</td>
<td>0.61</td>
<td>0.639</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.84</td>
<td>0.83</td>
<td>0.829</td>
<td>0.913</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.914</td>
<td>0.903</td>
<td>0.902</td>
<td>0.977</td>
</tr>
<tr>
<td>4</td>
<td>J48</td>
<td>0.852</td>
<td>0.845</td>
<td>0.844</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.773</td>
<td>0.585</td>
<td>0.499</td>
<td>0.704</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.715</td>
<td>0.714</td>
<td>0.713</td>
<td>0.735</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.846</td>
<td>0.841</td>
<td>0.841</td>
<td>0.925</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.917</td>
<td>0.913</td>
<td>0.912</td>
<td>0.974</td>
</tr>
<tr>
<td>6</td>
<td>J48</td>
<td>0.827</td>
<td>0.771</td>
<td>0.761</td>
<td>0.79</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.778</td>
<td>0.601</td>
<td>0.526</td>
<td>0.727</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.695</td>
<td>0.679</td>
<td>0.672</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.844</td>
<td>0.83</td>
<td>0.828</td>
<td>0.913</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.897</td>
<td>0.888</td>
<td>0.887</td>
<td>0.972</td>
</tr>
<tr>
<td>8</td>
<td>J48</td>
<td>0.861</td>
<td>0.839</td>
<td>0.836</td>
<td>0.867</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.786</td>
<td>0.626</td>
<td>0.566</td>
<td>0.741</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.725</td>
<td>0.719</td>
<td>0.717</td>
<td>0.785</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.883</td>
<td>0.866</td>
<td>0.865</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.914</td>
<td>0.908</td>
<td>0.907</td>
<td>0.971</td>
</tr>
</tbody>
</table>

Table 3: Link prediction accuracy (Precision, Recall, F-Measure, and Area Under Curve) of five supervised machine-learning classifiers over DBLP considering years 1995 through 2005 for training, and years 2006 through 2007 for testing. The tests were performed for author profiles of degree \( d \geq 1 \), \( d \geq 2 \), \( d \geq 4 \), \( d \geq 6 \), and \( d \geq 8 \).

### 4.6 Counting and algebraic analysis

Lastly, we perform counting, the most common and useful kind of analysis over DBLP. To do so, we consider the network as a bipartite graph with sets authors and articles in which the edges are time-stamped (year). Over this graph, we count two things for each author: **accomplishment** the number of distinct years when she/he published at least one article; and **silence** the biggest number of consecutive years without publishing. The first is an indicator of how active an author is, the bigger the accomplishment the more works were accomplished and the more productive is his career.

The second indicates how constant the author is, the bigger the silence, the less regular is her/his activity. Both metrics range from 0 to 50 years (nearly the longest academic career).

Figure 6 shows the histogram for **accomplishment** and **silence**. There one can see that the majority of DBLP’s authors is low-active with accomplishments between 1 to 4 years. Meanwhile, the histogram of Silence shows that a great share of DBLP is quite constant in what concerns their production regularity, with most authors having published something in the last year (silence 0) or in the years before (silence 1); however, a considerable share of authors silence periods between 2 and 20 years. Above 20 years we can safely consider that either the author has abandoned the academy or has passed away, in contrast to new (silence \( \leq 5 \)) active students.

We consider that these two metrics can provide an interesting characterization the overall profile of DBLP as they can combine to a single metric that translates to both productivity and constancy. In order to combine the two metrics we considered that they have inverted semantics: for one “more” is desirable, while for the other “less” is desirable, as depicted by the green and red arrows in Figure 6. Algebraically speaking, one is proportional and the other is inversely proportional, by combining both of them we got the metric **Sao Paulo’s Importance** as expressed by Equation 2.

In this equation, the logarithm of the accomplishment stands for the magnitude of the metric, and the square root of the silence penalizes the importance of a given author.

\[
S_{SP}^{‘\text{Importance}} = \frac{1}{\sqrt{\text{silence}} + 1} \ast \log(\text{Accomplishment})
\]

(2)

**Sao Paulo’s Importance**, or simply **Importance**, provides a number that translates how important a given author is; in the context of this work, importance refers to the insertion of edges in the network (publication and co-authoring), rather than to the relevance of articles. Having this metric in mind, we created a plot that figures the panorama of DBLP with respect to the hole of its authors. In Figure 7(a), it is possible to see the raw curve of metric **Sao Paulo’s Importance**; from the figure, it is possible to understand the behavior of the metric, which favors low silence and high accomplishment – reddish regions (high importance) as highlighted with a circle. In Figure 7(b), we present the counting (histogram) of authors per pair of Silence and Accomplishment – we use the same color mapping of figure (a), with high importance expressed in reddish colors.

This figure expresses how the **Sao Paulo’s Importance** is instantiated in DBLP; just a few authors (reddish region indicated by arrow) have high importance, with the great majority presenting low importance. It is interesting to see that there is a great share of authors with low silence (\( \leq 5 \) years) and low accomplishment (\( \leq 5 \) years) – this specific region defines a pick at the left-hand lower corner of the plot. Possibly, these authors refer to students that are still doing their PhD course, or that have recently finished it. This finding reveals how Computer Science is dependent on

\( ^{\text{in reference to the state of Sao Paulo in Brazil, which hosted this research.}} \)
casual researchers, and also how competitive it is, since just a few authors are able to migrate to the more important region of the plot.

Figure 7: Plot of metric Sao Paulo’s Importance. (a) Raw curve of Equation 2. (b) Counting (3D histogram) of authors in relation to the possible values of metric Sao Paulo’s Importance.

5. CONCLUSIONS
We introduced a multimodal analytical process defined as an ensemble of statistical (degree, and weakly-connected components distribution), topological (average clustering coefficient, and effective diameter evolution), algorithmic (link prediction/machine learning), and algebraic techniques to reveal non-evident features of network-like data, including networks of co-authoring, recommendation, computer routing, social interaction, protein interaction, to name a few. We demonstrated our process over the DBLP repository of Computer Science publications pointing out critical discoveries about its behavior from multiple perspectives.

Our methodology introduces an innovative course of action based on techniques that, although apart, can be used in complementary fashion. This kind of analytical approach is challenging due to its demand for heterogeneous technical knowledge and due to the diversity of the outputs to interpret. Nevertheless, it demonstrated a relevant potential in the form of ample interpretations of DBLP. These interpretations, in turn, can bring light to the research activity, possibly assisting in the decision making of funding agencies and academic personnel.

Acknowledgments
This work received support from Conselho Nacional de Desenvolvimento Científico e Tecnológico (560104/2010-3), and Fundação de Amparo a Pesquisa do Estado de São Paulo (2011/13724-1, 2013/03906-0, 2013/10026-7).

6. REFERENCES
Técnicas de análise de grafos aplicadas sobre a DBLP
Supervised-Learning Link Recommendation in the DBLP co-authoring network

Instituto de Ciencias Matematicas e de Computacao
Universidade de Sao Paulo
Avenida Trabalhador Sao-carlense, 400 - Centro, Sao Carlos, SP 13560-970, Brazil
Email: ggimenes@icmc.usp.br, gualdron@icmc.usp.br, raddo@usp.br, junio@icmc.usp.br

Abstract—Currently, link recommendation has gained more attention as networked data becomes abundant in several scenarios. However, existing methods for this task have failed in considering solely the structure of dynamic networks for improved performance and accuracy. Hence, in this work, we present a methodology based on the use of multiple topological metrics in order to achieve prospective link recommendations considering time constraints. The combination of such metrics is used as input to binary classification algorithms that state whether two pairs of authors will/should define a link. We experimented with five algorithms, what allowed us to reach high rates of accuracy and to evaluate the different classification paradigms. Our results also demonstrated that time parameters and the activity profile of the authors can significantly influence the recommendation. In the context of DBLP, this research is strategic as it may assist on identifying potential partners, research groups with similar themes, research competition (absence of obvious links), and related work.

I. INTRODUCTION

In the last decade, advances in the World Wide Web have led to improved mechanisms for users to interact and to share experiences, both for the general public and for corporations (industry and academy). Most of these social interactions are dynamics, receiving or loosing vertices and edges [1]. The dynamism of networks is itself a source of valuable, though not obvious, information; understanding such dynamism involves several variables that pose a complex problem [2]. This problematic has been dealt by several subfields, as graph theory, complex networks, and social network analysis (SNA); similar areas that differ by some subtleties. For the rest of this paper, we pick SNA as our area of concentration.

SNA refers to techniques and paradigms among which link recommendation (also known as link prediction) is of special interest [3]. Link recommendation refers to algorithmically foreseeing/identifying new associations between the existent vertices of a network; it is based on the assumption that the past and the present behavior of the net can indicate what may happen in the future. Such mechanism helps, for example, in problems like forecasting the behavior of a terrorist network [4]; in biology, it is used to identify associations that, otherwise, would demand intense experimentation to be discovered [2]; and, also, it is used in several kinds of social networking to expand the interaction among individuals.

One of the main paradigms of link recommendation is machine supervised learning, which is based on three different approaches [5]: the topological structure of the network, the semantic similarity among the properties of the vertices, and the description of the network behavior by means of probabilistic models [6]. Specifically, in this work, we use the topological structure of the network in order to recommend links by considering eight edge-oriented metrics based on neighborhood, path distance, and clustering properties [7]. We use these metrics in combination with a vast set of machine learning algorithms, presenting a comparative study that evaluates their relative efficiency.

We run experiments over the Digital Bibliography & Library Project (DBLP), a public database of Computer Science publications that defines a co-authorship graph. Link recommendation, in this sense, refers to identifying potential co-authoring (research collaboration) given previous and current co-authoring patterns. In the context of DBLP, the link recommendation that we propose is useful in identifying potential partners, research groups with similar themes, research competition (absence of obvious links), and related work. A recommended link does not necessarily mean that the correspondent authors should work together; rather, it is an indication that they should pay attention one to each other. For this task, we use machine learning classification algorithms; in our dynamic problem setting, the pairs of vertices are classified as positive or negative according to the edges that are created, or not, between their respective vertices. We considered techniques [8] J48, Naïve Bayes, Multilayer Perceptron, Bagging, and Random Forest, all of them available in the Weka framework, developed by the University of Waikato [9].

Specifically, our contribution is the use of supervised machine learning classification in the task of link recommendation in temporal graphs, proposing a systematic approach for computation and evaluation considering the time of publications and the profiles (number of publications) of the authors.

Following, we present works related to our proposal in Section II and the formalization of our methodology in Section III. In Section IV, we describe a vast set of experiments whose results are discussed in Section V. Section VI concludes the paper.

II. RELATED WORKS

Liben-Nowell and Kleinberg [10] present one of the most important works on link prediction/recommendation; the authors formalize the link prediction problem as the question of whether it is possible to infer which new interactions are likely to occur given a snapshot of a social network. They
use an unsupervised learning approach and, by calculating
similarity measures, they create a ranking by descending order
of similarity. The ranking is then used to recommend the
interactions that are likely to occur, in such a way that the
higher the rank, the more likely the interaction is to appear
in the future. The authors also acknowledge that the results
of using such ranking are not satisfactory and propose that
other approaches should be explored. In light of this matter,
we take a different direction by considering supervised learning
methods.

Recently, Aiello et al. [11] describe how friends that have
similar profiles (homophily) tend to get interconnected. In
their study, the authors consider the groups to which the
users belong, and the annotations (tags) of the users, among
other features. With these features, the authors calculate the
similarity between users, proposing a similarity threshold to
state whether two users are to define a connection, or not. Re-
gardless of its significative results, this study extrapolates the
topological information of the network; it relies on information
that, often, is not available or is not well-defined. This same
limitation is faced by Brandao et al. [12] and Lim et al. [13].

Clauset et al. [14] present the link prediction problem
based on a hierarchical analysis approach. Their method not
only provides interesting results for link recommendation, but
also explains many characteristics of the network. Despite its
results, their work demands that the graph representation be
hierarchically partitioned, a requirement that adds up com-
xplexity and processing demands; the same characteristic is
observed in Guo et al. [15]. In this work, we use a simpler,
yet efficient, method to accomplish link recommendation with
similar potential.

Zhou et al. [16] firstly evaluated how metrics that are
exclusively topological can be used for link recommendation.
In their work, the authors compare the performance of local
and global metrics. They conclude that local metrics, as used
in our work, is the better choice. However, different from
our approach, they consider the sole metrics instead of their
combination for improved performance. In [17], Papadimitriou
et al. use global graph processing in order to recommend
friends in social networks; although they achieved good results,
the technique is computationally expensive.

Menon et al. [18] analyses the effectiveness of matrix fac-
torization techniques for the structural link prediction problem.
They discuss a novel mechanism to allow their model to over-
come the imbalance characteristic using the idea of optimizing
for a ranking loss. Their results show good performance related
to the imbalance overcome. Finally the authors suggest that
the model can be used in conjunction with other approaches
to further improve the technique.

Sa and Prudencio [19] addresses link prediction by means
of classification algorithms and edge-oriented metrics; al-
though their work evaluates the role of edge weighting in the
task of foreseeing new links, they do not consider the multiple
parameters that influence the problem. The same approach is
used by Herman et al. [20]. With a different orientation,
we obtain better recommendation results that are discussed in
light of empirical experiments considering a wider spectrum of
configurations.

In this work, we present a methodology that extends former
proposals by defining a topology-exclusive link recommenda-
tion, with low complexity and processing cost, considering the
combination of multiple metrics in a comparative context. We
perform experiments that reveal how the different parameters
of the link recommendation problem affect diverse classifica-
tion algorithms. Our result is a method that reaches superior
rates (≈ 90%) of recommendation accuracy and the same
time that it indicates what are the most effective classification
algorithms for link prediction and recommendation.

### III. Methodology

In terms of a co-authorship graph, we have $G = (V, E)$, where $V$ is the set of vertices (authors), $E$ is the set of edges, so that
each edge $e = (u, v) \in E$ represents the co-authoring between
authors $u$ and $v$. Also, since we are worried about the dynamic
behavior of the network, each edge $e$ has a time label $t$ that
states when the edge was created; from the DBLP dataset, we
are considering the snapshot $1974 \leq t \leq 2007$. In this work,
given a snapshot of a network at time $t'$, we are interested in
recommending the edges that most likely should/could exist in
time $t''$, $t' < t''$; but that, for some reason, are still latent.

For link recommendation, we shall use the past and the
present behavior to recommend prospective new edges. There-
fore, it is necessary to break the set $E$ into two disjoint
subsets according to the time labels, defining past and present
intervals of time. The first interval – the past, is delimited
by two moments $t_1$ and $t_2$, $t_1 < t_2$, and is used as the
training interval, which we refer to as the induced subgraph
$G[t_1, t_2] = (V, E_{past})$. The second interval – the present,
is delimited by other two moments, $t_3$ and $t_4$, $t_3 < t_4$, which we
refer to as $G[t_3, t_4] = (V, E_{pres})$.

### TABLE I. METRICS USED IN THIS WORK.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of common neighbors (CN)</td>
<td>$CN(x,y) =</td>
</tr>
<tr>
<td>Jaccard’s coefficient (IC)</td>
<td>$JC(x,y) = \frac{</td>
</tr>
<tr>
<td>Preferential attachment (PA)</td>
<td>$PA(x,y) = \frac{</td>
</tr>
<tr>
<td>Adamic-Adar coefficient (AA)</td>
<td>$AA(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log</td>
</tr>
<tr>
<td>Path distance (PD)</td>
<td>Shortest path between $x$ and $y$</td>
</tr>
<tr>
<td>Resource allocation index (RA)</td>
<td>$RA(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{</td>
</tr>
<tr>
<td>Local path (LP)</td>
<td>$LP(x,y) =</td>
</tr>
<tr>
<td>Local clustering coefficient (CC)</td>
<td>$ANCC(x,y) = cc(x) + cc(y)$</td>
</tr>
</tbody>
</table>

Given a co-authorship graph $G$, link recommendation be-
comes a two-class problem to be treated with classification
techniques – positive instances refer to pairs of vertices (po-
tential links) that could be connected in the future, and negative
ones refer to the other case. In order to be classified, the pairs
must be represented as vectors of numbers; in this case, each
dimension of the vectors is a metric. We use edge-oriented
metrics calculated straightforward from the topological information
of the network. The advantage of such metrics is their domain-
independence because they can be calculated from any kind of
network. In Table I, we present the metrics that we use – for
these metrics, we consider the following definitions: let $\Gamma(x)$
be the set of neighbors of vertex $x$; $|\Gamma(x)|$ be the degree of $x$;
and $e(x, y)$ be the non-directed edge between $x$ and $y$.

The classifiers we employ – J48, Naïve Bayes, Multilayer

564
Perceptron, Bagging, and Random Forest, see Table II, learn from the past of the network, which is represented as pre-classified vectors corresponding to the pairs of vertices; these pairs are classified according to what is observed in the present of the network. In our experiments, the classifiers use 10% of the present information to pre-classify the vectors (pairs), using this data to lean and recommend the remaining 90% of the present data. Therefore, the accuracy corresponds to how precise the recommendations match the known 90% of the present. We use the classical 10-fold cross-validation, that is, we perform the same classification 10 times, each one using only 10% of what is already known about the data. The final performance is given by the average of the results.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Details</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>Decision tree algorithm</td>
<td>-C 0.25 -M 0.2</td>
</tr>
<tr>
<td>Naïve Bayes (NB)</td>
<td>Probabilistic neural network</td>
<td>-L 0.3 -M 0.2 -N 0.5 -V 0 -S 0 -E 20 -H a</td>
</tr>
<tr>
<td>Multilayer Perceptron (MLP)</td>
<td>Meta-classifier</td>
<td>-P 100 -S 1 -I 10 -W ADTree -B 10 -E -3</td>
</tr>
<tr>
<td>Bagging</td>
<td>Combination of decision trees</td>
<td>-I 10 -K 3 -S 1</td>
</tr>
<tr>
<td>Random Forest (RF)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE II. CLASSIFIERS USED IN THIS WORK.

Figure 1 presents the general flow of the link prediction task, (1) shows the extraction of the topological features from the DBLP network, (2) represents the training of the classifiers that we use, (3) is the classification process itself in which the new links are recommended, number (4) presents the evaluation step where measures like AUC are used to quantify the efficiency of the classifiers, finally (5) consists of the analysis of the results obtained and can be considered the end of the task, from the results knowledge can be obtained in such a way that it is possible to learn from the behavior of the network and use it to predict and recommend new links.

![Flowchart](image.png)

Fig. 1. Link recommendation task overview.

In dynamic graphs, there are vertices that remain active throughout the life span of the network, and there are vertices that simply pop out and become inactive right after. Therefore, another aspect is which vertices to consider for recommendation. To this end, we use the number of coauthorings (the degree) of the authors as criterion; we consider only the authors that have at least \( k \) coauthorings in both the past and in the present intervals. The set of vertices that satisfies the criterion for a given \( k \) is denoted core of authors. In our experiments, we discuss parameter \( k \) empirically by evaluating different values for it.

### IV. Experiments

We have run the link recommendation experiment considering three different time settings; using classifiers J48, Naïve Bayes, Multilayer Perceptron, Bagging, and Random Forest; and considering \( k \in \{1, 2, 4, 6, 8\} \).

#### Time settings

For the DBLP snapshot, whose years range from 1974 to 2007, we consider the following intervals for the past and the present of the network – see Section III.


Each time setting had a different core of authors depending on parameter \( k \). In the following, we analyze the resulting core for each setting considering \( k \in \{1, 2, 4, 6, 8\} \). For the first time setting, \( G[1995, 2005] \) – \( G[2006, 2007] \), the value of \( k \) is induced a core reduction of 98%; from 512,929 authors to only 7,583 – see Table III. The reduction eliminated authors that do not have at least one edge either in the past, and/or in the present, as defined in Section III. For \( k = 2 \), the reduction was still significative, 77% less authors; the same holds for \( k = 4 \), with a reduction of 51%. For higher values, \( k = 6 \) and \( k = 8 \), the reduction was less intense – around 10%. The same behavior is observed for the second and third time settings. The observations indicate that the majority of the authors are eventual researchers with one or two publications, and also that there are very few researchers with a constant and high (> 4) number of publications. These results suggest that the value of \( k \) must be between 2 and 4.

<table>
<thead>
<tr>
<th>( k )</th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7583</td>
<td>25781</td>
</tr>
<tr>
<td>2</td>
<td>1714</td>
<td>9458</td>
</tr>
<tr>
<td>3</td>
<td>826</td>
<td>5937</td>
</tr>
<tr>
<td>4</td>
<td>760</td>
<td>5681</td>
</tr>
<tr>
<td>5</td>
<td>756</td>
<td>5654</td>
</tr>
<tr>
<td>6</td>
<td>266877</td>
<td>57677</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( k )</th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1056</td>
<td>6627</td>
</tr>
<tr>
<td>2</td>
<td>569</td>
<td>4940</td>
</tr>
<tr>
<td>3</td>
<td>530</td>
<td>4758</td>
</tr>
<tr>
<td>4</td>
<td>529</td>
<td>4751</td>
</tr>
<tr>
<td>5</td>
<td>2657</td>
<td>2495</td>
</tr>
<tr>
<td>6</td>
<td>2657</td>
<td>2748</td>
</tr>
</tbody>
</table>

TABLE III. DBLP TIME SETTINGS AND CARDINALITY OF THE CORE OF AUTHORS FOR \( k \in \{1, 2, 4, 6, 8\} \).

#### Processing

We used and extended the Stanford Network Analysis Project (SNAP) library to calculate the metrics needed in our methodology. For each time setting and value of \( k \), in a total of 15 configurations, we calculated the topological metrics considering...
only the vertices that satisfy to the core of authors definition. Algorithm 1 shows the pseudo code used to calculate the metrics in our experiment.

Algorithm 1 Algorithm to calculate the metrics of a co-authoring network.

Input: G: graph adjacency list
Input: t1: start of the first interval
Input: t2: start of the second interval
Input: k: Core of authors parameter
Input: N: number of pairs to consider

begin
  Graph \( G_{it1} = \text{InduceIntervalSubgraph}(G, t1, t1 - 1); \)
  Graph \( G_{it2} = \text{InduceIntervalSubgraph}(G, t2, G.\text{getLastYear}(v)); \)
  for each \( v \in (G_{it1} \cap G_{it2}) \) do
    if \((G_{it1}.\text{getDegree}(v) < k) \) OR \((G_{it2}.\text{getDegree}(v) < k) \) then
      \( G_{it1}.\text{delete}(v); \)
      ClassifyEdgesOf\((v, G_{it1}, G_{it2}); //Edges appearing in \( G_{it2} \) are positive
    end
  for \( i = 1 \) to \( N \) do
    \( c = G_{it1}.\text{graphSearchRandomPositiveEdge}(); \)
    WriteFile\(\text{CalculateMetricsFor}(c)\);
    \( c = G_{it1}.\text{graphSearchRandomNegativeEdge}(); \)
    WriteFile\(\text{CalculateMetricsFor}(c)\);
  end
end

Our algorithm receives 4 parameters, besides the graph that represents the network. The first two define the past and the present of a time setting, the third one refers to parameter \( k \), and the last one sets the number of edges to consider in the experimentation; since the number of possible edges (pairs) is very high \(((|V|(|V| - 1))/2\)), we randomly choose the pairs using graph search; that is, by traversing existing edges, or by jumping to random vertices.

Analysis of the metrics

Following, we empirically analyze the metrics calculated with Algorithm 1 for the first time setting. In Figure 2, we present the distribution of the values of each metric considering 400 positive examples, and 400 negative examples. One can see that metrics Number of common neighbors (a), Jaccard’s coefficient (b), Preferential attachment (c), Adamic-Adar coefficient (d), Resource allocation index (e), Preferential attachment (f), Local path (g), and Local clustering coefficient (h).

![Fig. 2. Distribution of the values of the metrics used in this work; dark gray for positive, and light gray for negative examples. Number of common neighbors (a), Path distance (b), Jaccard’s coefficient (c), Adamic-Adar coefficient (d), Resource allocation index (e), Preferential attachment (f), Local path (g), and Local clustering coefficient (h).](image)

V. RESULTS AND DISCUSSION

The experimental setting allowed us to observe the influence of the parameters (time setting, value of \( k \), and classifier) of the methodology. We present the results in Tables IV, V, and VI; and in the corresponding Figures 3, 4, and 5. Each table/figure corresponds to one time setting, including data for each classifier and \( k \) value, and presents evaluation measurements Precision, Recall, F-Measure, and Area Under Curve (AUC) corresponding to the Receiver Operating Characteristic (ROC). The numbers refer to the average results achieved with a 10-fold cross-validation. In the evaluation, the higher the values, the more trustworthy are the recommendations. Even the recommendations that did not match a future link are of interest; they can be interpreted as potential interactions that have not occurred; or interpreted as concurrent authors, in the case of rivalry.

In the three tables, the Random Forest (RF) classifier presented the highest scores in the three time settings, for all the values of \( k \), and considering the 5 classifiers. Classifier RF
uses random combinations of the metrics to produce multiple decision trees that will be merged by voting; that is, the classification given by the bigger number of trees is the final classification. It generates combinations that disregard specific metrics one at a time, a course of action that allowed RF to be less sensible to the peculiarities of individual metrics. Empirically, we verified that the classification was very effective ($\approx 90\%$ accuracy), especially if compared to the other classifiers that necessarily depend on all the metrics.

In contrast to classifier RF, classifiers Multilayer Perceptron and Naïve Bayes demonstrated to be inadequate for the link recommendation task. In Figures 3, 4, and 5, it is evident that their recommendation potential is irregular and pronouncedly inferior. We suspect that the number of metrics and their strong non-linear separability posed hard challenges to these classifiers that, differently from the other three, are not decision-tree based. On the other hand, the fact that there are only two classes possibly led to the improved performance of J48, Bagging, and Random Forest.

The results also demonstrated that for $k \in \{2, 6\}$ the link recommendation had a higher performance. This observation corroborates our initial guess, according to which we should have better results with profiles of around 4 publications per period (past and present), discarding left-most and right-most outliers. These results are observed for the three time settings, but they are more evident for the third time setting – Table IV and Figure 3. It is an indication that the link recommendation practice has a bigger potential for short past and short present configurations, situations in which the memory of the system is more recent and can better explain the near future.

The results presented in Tables IV, V, and VI indicate around $90\%$ of efficiency and accuracy. This rate is comparable to the works presented in Section II, being superior for DBLP and for similar datasets.

**VI. CONCLUSIONS**

We have touched the problem of link recommendation in the context of research collaboration over the DBLP dataset.

<table>
<thead>
<tr>
<th>k</th>
<th>Classifier</th>
<th>PRECISION</th>
<th>RECALL</th>
<th>F-MEASURE</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>J48</td>
<td>0.84</td>
<td>0.813</td>
<td>0.809</td>
<td>0.834</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.791</td>
<td>0.64</td>
<td>0.586</td>
<td>0.752</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.749</td>
<td>0.741</td>
<td>0.739</td>
<td>0.786</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.851</td>
<td>0.84</td>
<td>0.839</td>
<td>0.819</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.891</td>
<td>0.888</td>
<td>0.887</td>
<td>0.957</td>
</tr>
<tr>
<td>2</td>
<td>J48</td>
<td>0.857</td>
<td>0.838</td>
<td>0.835</td>
<td>0.855</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.785</td>
<td>0.655</td>
<td>0.661</td>
<td>0.762</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.718</td>
<td>0.715</td>
<td>0.714</td>
<td>0.777</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.854</td>
<td>0.84</td>
<td>0.841</td>
<td>0.891</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.925</td>
<td>0.915</td>
<td>0.915</td>
<td>0.971</td>
</tr>
<tr>
<td>4</td>
<td>J48</td>
<td>0.883</td>
<td>0.878</td>
<td>0.877</td>
<td>0.914</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.781</td>
<td>0.626</td>
<td>0.567</td>
<td>0.798</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.77</td>
<td>0.768</td>
<td>0.767</td>
<td>0.851</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.872</td>
<td>0.865</td>
<td>0.864</td>
<td>0.922</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.93</td>
<td>0.924</td>
<td>0.923</td>
<td>0.974</td>
</tr>
<tr>
<td>6</td>
<td>J48</td>
<td>0.842</td>
<td>0.824</td>
<td>0.821</td>
<td>0.877</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.764</td>
<td>0.64</td>
<td>0.592</td>
<td>0.767</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.799</td>
<td>0.783</td>
<td>0.779</td>
<td>0.804</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.848</td>
<td>0.84</td>
<td>0.839</td>
<td>0.924</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.917</td>
<td>0.908</td>
<td>0.907</td>
<td>0.968</td>
</tr>
<tr>
<td>8</td>
<td>J48</td>
<td>0.823</td>
<td>0.815</td>
<td>0.814</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>0.753</td>
<td>0.62</td>
<td>0.563</td>
<td>0.761</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>0.696</td>
<td>0.695</td>
<td>0.695</td>
<td>0.789</td>
</tr>
<tr>
<td></td>
<td>Bagging</td>
<td>0.853</td>
<td>0.849</td>
<td>0.848</td>
<td>0.914</td>
</tr>
<tr>
<td></td>
<td>RF</td>
<td>0.921</td>
<td>0.914</td>
<td>0.913</td>
<td>0.958</td>
</tr>
</tbody>
</table>

Fig. 5. AUC visualization of the data generated for the third time setting – $G^{[1995,1999]}, G^{[2000,2004]}$.

Acknowledgments

This work received support from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq-56104/2010-3), Fundacao de Amparo a Pesquisa do Estado de Sao Paulo (FAPESP-2011/13724-1, 2013/03906-0, 2013/10026-7) and Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior (Capes).

References

Sumarização visual de grafos
Large Graph Analysis in the GMine System

Jose F. Rodrigues Jr., Member, IEEE, Hanghang Tong, Member, IEEE, Jia-Yu Pan, Agma J.M. Traina, Senior Member, IEEE, Caetano Traina Jr., Senior Member, IEEE, and Christos Faloutsos, Member, IEEE

Abstract—Current applications have produced graphs on the order of hundreds of thousands of nodes and millions of edges. To take advantage of such graphs, one must be able to find patterns, outliers, and communities. These tasks are better performed in an interactive environment, where human expertise can guide the process. For large graphs, though, there are some challenges: the excessive processing requirements are prohibitive, and drawing hundred-thousand nodes results in cluttered images hard to comprehend. To cope with these problems, we propose an innovative framework suited for any kind of tree-like graph visual design. GMine integrates 1) a representation for graphs organized as hierarchies of partitions—the concepts of SuperGraph and Graph-Tree; and 2) a graph summarization methodology—CEPS. Our graph representation deals with the problem of tracing the connection aspects of a graph hierarchy with sub linear complexity, allowing one to grasp the neighborhood of a single node or of a group of nodes in a single click. As a proof of concept, the visual environment of GMine is instantiated as a system in which large graphs can be investigated globally and locally.

Index Terms—Graph analysis system, graph representation, data structures, graph mining, graph visualization

1 INTRODUCTION

Large graphs are common in real-life settings: web graphs, computer communication graphs, recommendation systems, social networks, bipartite graphs of weblogs, to name a few. To find patterns in a large graph, it is desirable to compute, visualize, interact and mine it. However, dealing with graphs on the order of hundreds of thousands of nodes and millions of edges brings some problems: the excessive processing requirements are prohibitive, and drawing hundred-thousand nodes results in cluttered images hard to comprehend.

In former works, the large graph problem has been treated through graph hierarchies, according to which a graph is recursively broken to define a tree of sets of partitions. However, previous efforts on this matter fail on the task of integrating the information from multiple partitions, disregarding mining techniques to fine inspect each subgraph. Conversely, for understanding a graph hierarchy, it is worthwhile to have systems that provide aids for answering the following questions:

- Hierarchical navigation: What is the relation between arbitrary groups (partitions) of nodes?
- Representation and processing: What are the adjacencies of a given graph node considering the entire graph, and not only its particular partition?
- Mining: Given a subset of nodes in the graph, what is the induced subgraph that best summarizes the relationships of this subset?
- Visualization: How do we see through the levels of the graph hierarchy?
- Interaction: How do we perform all these tasks efficiently and intuitively?

It is our contention that a system that presents the original graph concomitant to its hierarchical version must meet all these requirements. Therefore, we seek for a new representation for graph hierarchies, different from previous works in which the graph hierarchy is “stagnant” and cannot answer questions about the relationships between nodes at different groups, and neither between groups at different partitions of the hierarchy. These are serious limitations because a graph is, essentially, a model for representing relationships.

Another concern is that even at the deepest level of a graph hierarchy—at the leaves, it is possible to find subgraphs complex enough to surpass the analytical capacity. In this situation, one should be able to summarize the subgraph achieving a small, yet representative, fraction of it; an operation that answers for a deeper level of insight over hierarchical partitionings.

The contribution of this work is the integration of methodologies that address the problems discussed above. We introduce a novel representation for graph hierarchies that extends those of previous works, leading to a model more suitable for presentation and computation. Our methodology also counts on the possibility of graph summarization at the subgraphs (leaves) in a graph hierarchy. The result of our efforts is GMine [32], a system that allows browsing and mining of large graphs in a rich visual environment [26].


For information on obtaining reprints of this article, please send e-mail to tkde@computer.org, and reference IEEECS Log Number TKDE-2011-02-0063.

Digital Object Identifier no. 10.1109/TKDE.2011.199.
The paper is organized as follows: Section 1 reviews related works for this paper. Section 3 introduces the SuperGraph/Graph-Tree methodology and Section 4 explains the CEPS graph summarization. Section 5 presents experiments on the Graph-Tree performance and Section 6 presents accuracy measures for CEPS. As a proof of concept, Section 7 demonstrates the GMine system. Section 8 concludes the paper.

2 Related Work

The interest on large graph analysis has increased in the recent years. This research area includes pattern mining [10], influence propagation [18] and community mining [15], among others. Such themes can benefit from tools that enable the visual inspection of large graphs.

2.1 Graph Hierarchical Presentation

Although many works implicitly define the hierarchical clustering of graphs—as in the work of Eades and Feng [11], most of them do not touch the issue of how such arrangements deal with scalability and processing by means of a well-defined data structure. Batagelj et al. [6], for instance, generalizes on the concept of X-graph of Y-graphs to define a properties-oriented hierarchical clustering of graphs, but not providing details nor performance evaluation of the implicit data arrangement that supports their processing. Archambault et al. [4] define an ingenious dynamic modification of the graph hierarchy in light of a single node of interest; their system requires the user to reset her/his referential focus at every new choice of a node with a strictly linear complexity on the basis of seconds delay. Gansner et al. [16] present a fish-eye visualization built over a graph layout with precomputed coordinates, their structure permits the inspection of the graph at multiple levels of details. Schaffer et al. [33] describe an earlier fish-eye approach focused on the interactive experience. From the aesthetic perspective, van Ham and van Wijk [20] present an interesting technique to visualize small-world graphs using interactive clustering and an enhanced force-directed algorithm [12]. Auber et al. [5] present a work on the same theme using the clustering index metric [23]. For the problem of nonclustered drawing, Harel and Yehuda [21] describe an efficient method based on the embedding of graphs in high-dimensional spaces followed by a Principal Component Analysis (PCA) dimensionality reduction to two or three dimensions.

Huang and Nguyen [22] present a methodology for visualizing hierarchical graphs. They introduce an efficient layout scheme, being able to scale to tens of thousands of nodes. Different from our work, they do not integrate the relationships lost after the hierarchy generation; neither do they use a proper data structure, so their system is limited to main memory. Papadopoulos and Voglis [30] propose a drawing method based on graph modular decomposition [8]. Their work does not present a complete system, but a description of how to arrange the modules of a graph according to hierarchical levels. In the GrouseFlocks system, Archambault et al. [3] define metanodes and metaedges to introduce the same visualization paradigm that we employ in our proof of concept experiments; differently, they focus on layout and interaction with one order of magnitude higher processing demands for smaller graphs. Generally, former works—as those presented by Finocchi [14]—have not considered the issue of efficiently managing graph hierarchies, instead, they rely on ad hoc linear or matrix adjacency structures. The use of such structures leads to hierarchies that do not provide comprehensive graph relationship information, mostly due to the scalability shortcomings of these approaches. In the literature, the goal of authors has been aesthetics; while here, we aim at a model that is more suitable for large scale computation and mining.

In the specific field of hierarchical graph navigation, Buchsbaum and Westbrook [7] formally present the problem and provide a solution in which the graph hierarchy has one unique associated state that changes according to two possible transitions: expand and contract. In their model, the graph nodes and the nodes of the hierarchy are a single concept at different levels of abstraction. In another work, Raitner [31], along with an extensive research compilation, deals with the issue of dynamically editing the nodes that are under a subtree of the hierarchy structure. These two works are references for what is known as graph view maintenance problem. Differently to the view maintenance approach, we describe a framework that aims not only at hierarchical navigation, but at large graph processing by means of a data structure that can fully represent a graph by abstracting the fact that it is hierarchically partitioned. Our structure is based on three integrated concepts: graph hierarchy, subgraphs, and graph nodes; it can restore the adjacency information of a single graph node or compute the relationship of arbitrary graph partitionings with a fraction of the original graph in memory, defining a complete graph representation over a hierarchical structure.

In speaking about visual design, the field of tree-like visualization is long-term now and has a great number of branches as compiled by Hans-Jörg Schulz at http://treevis.net/. In this scenario, the aim of our work is to propose processing techniques that fit to any tree-like design in the task of scalable hierarchical graph visualization. As so, GMine’s visual appeal was conceived as a proof of concept of our intent, accordingly, it does not compete with more elaborated designs.

2.2 Graph Representation

Two classic data structures usually are used for graph representation: adjacency matrices and adjacency lists. Another possibility is to use Binary Decision Diagrams [2], which represent the nodes of the graph using binary sequences. This approach supports massive processing using less memory, however, the nodes can no longer be processed individually [17]. These three techniques are limited to main memory, this is because they are plain and do not provide the benefits of optimized disk access offered by hierarchical structures. Another line of research considers out-of-memory algorithms [37], according to which the graph is preprocessed for specific computations. Such algorithms minimize disk accesses, however the computation is not versatile and does not favor interaction. Finally, Davi et al. [9] define a representation for hierarchically partitioned graphs similar to our approach—using the concepts of SuperNodes and SuperEdges; however, their representation is intended for completely different purposes—keyword search over graphs.
2.3 Graph Summarization

Besides the capability of globally analyzing large graphs, our system is complemented with the possibility of locally analyzing a subgraph that is part of a larger graph hierarchy. For this aim, we use a graph summarization method named Center-Piece Subgraph—CEPS [36]—adapted for visual interaction and presentation, and embedded at the leaves of our graph representation. A center-piece subgraph contains the collection of paths connecting a subset of nodes of interest. It has been shown that the center-piece subgraph can discover a collection of paths rather than a single path, and is preferable to other methods on describing the multifaceted relationship between entities in a social network. The CEPS method uses random walk with restart (RWR) to calculate an importance score between graph nodes. Random walks refer to stochastic processes where the position of an entity, in a given time, depends on its position at some previous time. There are many applications using random walk methods, including PageRank [28], cross-modal multimodal correlation discovery [27], and neighborhood formation in bipartite graphs [34].

The MING approach [25] extends CEPS’ ideas to disk-resident graphs and to the Entity-Relationship database context providing the IRank measure to capture the informativeness of related nodes. In recent works, Patel et al. conduct a research effort on how to produce graph summaries. Their SNAP summarization uses node attributes combined to the implicit domain knowledge embedded in the graph structure and content [35]; further in this line [38], an automatic numerical categorization produces multiple summaries compared by means of a measure of interestingness.

CEPS also relates to the concept of “goodness” of a connection subgraph. The two most natural measures for goodness are the shortest distance and the maximum flow. However, as pointed out by Faloutsos et al. [13], both measurements fail to capture some preferred characteristics for social networks. A more related closeness (distance) function is proposed by Palmer and Faloutsos [29]. However, it cannot describe the multifaceted relationship that is essential in social networks. In [13], Faloutsos et al. propose a method based on electricity current, in which the graph is seen as an electric network. By applying +1 voltage to one query node and setting the other query nodes at 0 voltage, their method chooses the subgraph which delivers maximum current between the query nodes. The delivered current criterion can only deal with pair wise source queries, which is a special case of the CEPS graph summarization.

3 SuperGraphs and the Graph-Tree

Our first contribution is an original formalization of graph hierarchies engineered to support processing and presentation. We define the SuperGraphs concept, an abstraction that converges to an implementation model we have named Graph-Tree. While SuperGraphs formalize the essentials of the Graph-Tree, the Graph-Tree incorporates the SuperGraph abstraction. SuperGraphs extend previously proposed graph hierarchy representations—Section 3.4—while the Graph-Tree instantiates it in a way that is propitious for efficient computation—Section 5 and interactive presentation—Section 7.

The closest work to the ideas of SuperGraph and Graph-Tree was proposed by Abello et al. [1]. Their work formalizes a hierarchy tree, whose data structure is based on what they name antichains—sets of nodes such that no two nodes are ancestors of one another. Their formalization parallels with ours by the concept of macro—similar to the terminology super, used along this work. Their structure stores a static set of macro (super) edges between the macro (super) nodes of the hierarchy; differently, our data structure introduce the Connectivity computation, a dynamic means to determine macro (super) edges between arbitrary macro (super) nodes, even for the leaves (solely nodes). The originality of our approach is that the graph hierarchy is not available only for visual interaction; it can be used for processing at any level of the tree just as if the original graph was a thorough plain representation. This is possible due to the connectivity computation embedded in the Graph-Tree, as defined in Section 3.4.

3.1 Graph-Tree Structure Formalization

For the purpose of formalizing the Graph-Tree structure, following we define a set of abstractions that encompass its engineering, starting by the notion of SuperGraph. The underlying data beneath a SuperGraph are a graph \( G = \{V, E\} \)—with \(|V|\) nodes and \(|E|\) edges—but a SuperGraph presents a different abstract structure. It is based on the observation that the entities in a graph can be grouped according to the relationships that they define. This concept allows us to work with a graph as a set of partitions hierarchically defined. In the following, we define the constituents of a SuperGraph, illustrating them with the example in Fig. 1.

**Definition 1 [SuperGraph].** Given a finite undirected graph \( G = \{V, E\} \), with no loops nor parallel edges, a SuperGraph is defined as \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} \) is a set of SuperNodes \( \mathcal{V} \), \( \mathcal{V} \) is a set of LeafSuperNodes \( \mathcal{V} \), and \( \mathcal{E} \) is a set of SuperEdges \( \mathcal{E} \). In the following, we define LeafSuperNode, SuperNode, and SuperEdge.

---

1. For a standard formalism on clustered graphs, see the seminal work of David [19].
Definition 2 [LeafSuperNode]. Given a subset of graph nodes \( V' \subseteq V \), a LeafSuperNode \( \pi \) is defined as the subgraph \( G' = \{ V', E' \} \), where \( E' = \{ (u, v) | (u, v) \in E \text{ and } u, v \in V' \} \).

Definition 3 [SuperNode]. A SuperNode \( \pi \) is recursively defined as a set \( V \) of SuperNodes, or LeafSuperNodes, \( \pi \), plus a set \( E \) of SuperEdges \( \pi \).

\[
\pi = \{ V_1, V_2, \ldots, V_{|\pi|-1} \}, \\
E = \{ \pi_{ij} \subseteq V \}.
\]

where \( \pi \) can be either a SuperNode or a LeafSuperNode; the concept of SuperEdge, \( \pi \), is introduced later in the next section. Fig. 1 illustrates the concepts of SuperNode and LeafSuperNode.

Note that SuperNode and LeafSuperNode correspond to “nodes” in the hierarchy defined in a Graph-Tree. They are not to be confused with the individual graph nodes of the underlying graph.

3.2 Basic Definitions of the SuperGraph

The SuperGraph abstraction naturally lends to a novel tree-like model that we call Graph-Tree. Following, we present the basic operations for the Graph-Tree to work.

Definition 4 [Coverage of a SuperNode]. Given a SuperNode \( \pi = \{ V, E \} \), the coverage of \( \pi \) is given by the recursive definition:

\[
\text{Coverage}(\pi) = \begin{cases} V' & \text{if } \pi \text{ is a LeafSuperNode} \\
\bigcup \text{Coverage}(\pi_i) & \text{otherwise},
\end{cases}
\]

where \( \pi \in V \), \( 0 \leq i \leq |V'| - 1 \).

The coverage of a SuperNode corresponds to the graph nodes that comprehend its community. At the leaves, a community is a subgraph and, at the root, the community is the entire graph.

Definition 5 [Parent(s) of a SuperNode]. We refer to the parent of a SuperNode \( \pi \) as Parent(\( \pi \)) = \( \pi = \{ V', E' \} \) if \( \pi \in \pi \). We refer to the set of ancestors of a SuperNode \( \pi \) as the set Ancestors(\( \pi \)) = \{ \pi | \pi \in V \text{ and } \pi \in \text{Coverage}(\pi) \}. Similarly, two SuperNodes (or LeafSuperNodes) are siblings if they have the same parent SuperNode.

Definition 6 [SuperEdges]. A SuperEdge represents all the edges \( (u, v) \in E \) that connect graph nodes from a SuperNode \( \pi \) to graph nodes from SuperNode \( \pi \). A SuperEdge \( \pi_{ij} \) for a LeafSuperNode \( \pi_{ij} = \{ V_1, E_1 \} \) holds all the edges that interconnect graph nodes in the LeafSuperNode \( \pi_{ij} \), that is, all the edges in \( E_1 \). Formally, the SuperEdge between SuperNodes \( \pi \) and \( \pi \) is defined as follows:

\[
\text{SuperEdge}(\pi, \pi) = \pi_{ij} = \{ e = (u, v) | (u, v) \in E', \\
u \in \text{Coverage}(v_i) \text{ and } v \in \text{Coverage}(v_j) \}.
\]

Definition 7 [Weight of a SuperEdge]. The weight of a SuperEdge is the sum of the weights of its edges.

Definition 8 [Internal Edge]. Given a SuperNode (or a LeafSuperNode) \( \pi \), an edge \( e \) is called an internal edge of \( \pi \) if \( \text{source}(e) \in \text{Coverage}(\pi) \) and \( \text{target}(e) \in \text{Coverage}(\pi) \).

The internal edge \( e \) can be resolved within the coverage of \( \pi \). For simplification, given an edge \( (u, v) \), \( u = \text{source}(e) \) and \( v = \text{target}(e) \), even if the edges are undirected.

Definition 9 [External edge]. An edge \( e \) is called an external edge of \( \pi \) if \( \text{source}(e) \in \text{Coverage}(\pi) \) and \( \text{target}(e) \notin \text{Coverage}(\pi) \). The external edge \( e \) cannot be resolved within the Coverage of \( \pi \).

Definition 10 [Open node]. A graph node \( v \in \text{Coverage}(\pi) \) is called an open node of \( \pi \) if there exists an external edge \( e \) in the set of external edges of \( (\pi) \) where \( \text{source}(e) = v \). We denote the set of all the open nodes of a SuperNode \( \pi \) as OpenNodes(\( \pi \)).

With these basic definitions in mind, the engineering of the Graph-Tree can be better understood by tracing its process of construction, as presented in the next section.

3.3 Construction of the GraphTree

In this section, we describe how to build a Graph-Tree. We illustrate the process in order to clarify its structure and the information it manages.

3.3.1 Hierarchy Construction

The choice for a specific graph partitioning is independent of the Graph-Tree methodology. The partitioning can be part of a data set with a hierarchical structure, or it can be achieved via automatic partitioning. For automatic partitioning, in GMine, we recursively apply the k-way graph partitioning known as METIS, as described by Karypis and Kumar [24]. We perform a sequence of recursive partitionings. Each recursion generates \( k \) partitions to form the next level of the tree, a process that repeats until we get the desired number of \( h \) hierarchy levels. For each new set of partitions (subgraphs), new subtrees are embedded in the Graph-Tree. At the end of the process, references to the subgraphs are kept at the leaves. From the storage point of view, the tree-structure is kept on main memory, while the subgraphs are kept on disk, being read only when necessary.

3.3.2 Filling the Graph-Tree SuperNodes

After obtaining a hierarchy, it is necessary to fill the SuperNodes of the tree with their SuperEdge and open nodes information. In Algorithm 1, the Graph-Tree is recursively traversed bottom-up along its levels. Initially the LeafSuperNodes are filled with references to the subgraphs stored on the disk. Then, the algorithm proceeds to upper levels, where the external edges propagated from lower levels are used to resolve the SuperEdges and to track the open nodes.

Fig. 2 illustrates this process. We start with graph \( G \), which is partitioned to create the Graph-Tree with empty SuperNodes (see Figs. 2a, 2b, and 2c). The bottom-up recursive process starts at the leaves, illustrated in Fig. 2d. For this example, and for Fig. 2e, boldface indicates matches between external edges, while gray edges indicate unresolved external edges. Underlined graph node id’s indicate open nodes and the diagonal arrows depict the external edges propagated up the tree. Still in Fig. 2d, it is possible to see the information propagated from SuperNodes \( \pi_1 \) and \( \pi_2 \), which will be used in line 8 of Algorithm 1 to find matches between unresolved external edges. Fig. 2e illustrates the
crossing of the propagated data results in matches $(2, 3) – (3, 2)$ and $(2, 4) – (4, 2)$, stored in SuperEdge $\bar{e}_{3,4}$. Fig. 2e also shows the first SuperEdges among siblings, $(\bar{e}_{3,4}$ and $\bar{e}_{5,6})$. Fig. 2f shows the last SuperEdge storing the last set of edges between siblings. Fig. 2g shows the end of the process, when all the edges are spread along the data structure.

**Algorithm 1: Algorithm to fill a Graph-Tree.**

```plaintext
Input: Ptr: pointer to the root of the Graph-Tree
FillGraphTree(Ptr) begin
if P ptr is leaf then
    Set the variable P ptr → filePath to the file of the corresponding subgraph;
else
    for each child $s_i$ of P ptr do
        FillGraphTree($s_i$);
        /*Recursively down the hierarchy*/
    end
    Instantiate a SuperEdge for each pair of children;
    Find matches between the unresolved external edges from each pair of children;
    Store matching edges in the SuperEdges;
end
Use external edges to determine P ptr’s open nodes;
Propagate (unresolved) external edges to the parent;
end
```

### 3.4 SuperGraph Connectivity Computations

In this section, our aim is to answer the questions raised in Section 1 by dynamically restoring the original graph information.

#### 3.4.1 SuperNodes Connectivity

The connectivity between two SuperNodes in a hierarchy is the set of edges between them. For sibling SuperNodes, their connectivity corresponds to the SuperEdge that interconnects them, readily available as part of the SuperGraph. For SuperNodes that are not siblings, their connectivity must be traced.

**Definition 11 [SuperNodes Connectivity].** Given a SuperGraph $\mathcal{G} = (V, E)$ and two SuperNodes $\bar{v}_i$ and $\bar{v}_j$ in $\mathcal{G}$, the SuperNodes Connectivity for the pair $(\bar{v}_i, \bar{v}_j)$ is the set of edges

$$SNC(\bar{v}_i, \bar{v}_j) = \{e| source(e) \in Coverage(\bar{v}_i) \text{ and } target(e) \in Coverage(\bar{v}_j)\}.$$  

The challenge is how to trace the connectivity between arbitrary SuperNodes without having to cross the SuperGraph with the graph that originated it. To do so, we benefit from the SuperGraph definitions of the former section in order to calculate the connectivity between SuperNodes.

**Proposition 1 [All Possible Connecting Edges].** Given any two SuperNodes $\bar{v}_i$ and $\bar{v}_j$, the complete set of all possible edges connecting $\bar{v}_i$ to $\bar{v}_j$ is given by the Cartesian product $OpenNodes(\bar{v}_i) \times OpenNodes(\bar{v}_j)$.

**Proposition 2 [Connecting edges from the Common Parent].** The set of edges that connect any two SuperNodes $\bar{v}_i$ and $\bar{v}_j$ is a subset of the unique SuperEdge $\bar{e}_{gh}$ connecting SuperNodes $\bar{v}_g$ and $\bar{v}_h$, where $\bar{v}_g \in Ancestors(\bar{v}_i)$ and $\bar{v}_h \in Ancestors(\bar{v}_j)$, so that $\bar{v}_g = Parent(\bar{v}_i) = Parent(\bar{v}_j)$. Intuitively, $\bar{v}_g$ is the first common parent of $\bar{v}_i$ and $\bar{v}_j$, and $\bar{v}_g$ and $\bar{v}_h$ are sibling SuperNodes under $\bar{v}_g$ and are “ancestors” of $\bar{v}_i$ and $\bar{v}_j$, respectively.

From Propositions 1 and 2, it becomes possible to calculate the connectivity between two SuperNodes based on set operations, as follows:

**Proposition 3 [Computing SuperNodes Connectivity $SNC(\bar{v}_i, \bar{v}_j)$].** The set of edges $SNC(\bar{v}_i, \bar{v}_j)$ that connect any two SuperNodes $\bar{v}_i$ and $\bar{v}_j$ is the intersection between the set of all possible edges between $\bar{v}_i$ and $\bar{v}_j$ (Proposition 1) and the superset that contains (but not only) the set of edges between $\bar{v}_i$ and $\bar{v}_j$ (proposition 2). Formally, the SuperNodes Connectivity $SNC(\bar{v}_i, \bar{v}_j)$ is given by:

$$SNC(\bar{v}_i, \bar{v}_j) = \bigcap_{\bar{v}_g \in Coverage(\bar{v}_i), \bar{v}_h \in Coverage(\bar{v}_j)}\{OpenNodes(\bar{v}_g) \times OpenNodes(\bar{v}_h)\}.$$  

(4)

To see why Proposition 3 is the case, we note that $\bar{e}_{gh} = SuperEdge(\bar{v}_g, \bar{v}_h)$ contains all the edges between $Coverage(\bar{v}_i)$ and $Coverage(\bar{v}_j)$, and therefore it is a superset of $SNC(\bar{v}_i, \bar{v}_j)$.
3.4.2 Graph Nodes Connectivity (GNC)
A graph hierarchy stores groups (partitions) of nodes that are interrelated. However, the relationships between graph nodes at different groups are not stored; we lose information when we alter the graph representation. In a Super-Graph, it is possible to determine the relationships relative to any graph node, which we define as follows:

**Definition 12 [Graph Nodes Connectivity].** Given a Super-Graph \(G = (V, \overline{V}, E)\), a SuperNode \(\overline{v} \in \overline{G}\), and a graph node \(v \in \text{Coverage}(\overline{v})\), the Graph Nodes Connectivity for \(v\) (denoted as \(\text{GNC}(v)\)), is defined as the set of edges \(e \in E\) connecting \(v\) to all the other graph nodes that do not pertain to \(\overline{v}\). That is, \(\text{GNC}(v) = \{e \in E, \text{source}(e) = v \text{ and target}(e) \in \{V - \text{Coverage}(\overline{v})\}\}\).

**Proposition 4.** If a graph node \(v\) is an open node for a SuperNode \(\overline{v}\), then the set of ancestors \(\text{Ancestors}(\overline{v})\) have all the SuperEdges that hold edges connected to \(v\). Proposition 4 is a direct result from Definition 6.

Following Proposition 4, if we know the set of ancestors and the set of open nodes of a SuperNode, we can determine the relationships (external edges) of any graph node \(v \in \text{OpenNodes}(\overline{v})\). A reference to the immediate parent at each SuperNode is enough to define a recursive procedure to trace the external edges of any graph node \(v\). Such procedure checks each parent SuperNode, starting from the first parent above the leaves, up to the root. While \(v\) is in the set of open nodes of the parent SuperNode being checked, then there are still external edges to be traced.

In this section, we have presented the SuperGraph/Graph-Tree formalism, which carries an engineering that elegantly allows the construction of a graph hierarchy. It also predicts computation that can restore all the relationships of the original graph, and that can calculate relationships between SuperNodes at any levels of the hierarchy. In Section 5, we demonstrate that the Graph-Tree can perform its computations with sublinear complexity, scaling to graphs that are really big.

### 4 CEPS: Center-Piece Subgraph

Although graph hierarchies can lessen the problem of globally inspecting large graphs, we have found that it is common to reach the bottom of the Graph-Tree and have a subgraph that presents more information than what is desired, in a layout that suffers with node overlapping. In this situation, although the user is able to compute, draw, and interact with the graph nodes of a LeafSuperNode, there might still be too many edges and nodes, preventing examination. This happens naturally, either on large graphs or on moderate to small graphs.

To remedy this problem, we benefit from the concept of Center-Piece Subgraph (CEPS for short) to complement the analytical environment of GMine. A center-piece subgraph contains the collection of paths connecting a subset of graph nodes of interest. Using the CEPS method, a user can specify a set of query graph nodes and GMine will summarize and present their internal relationship through a small (say, with tens of nodes), yet representative connection subgraph.

CEPS aids on interaction by significantly reducing the number of edges and of nodes to be inspected; we can estimate its benefits analytically. For a complete graph \(G\)—a worst case situation—Fig. 3a, one must manually check \(|N|(|N| - 1)/2\) edges in order to manually generate a center-piece subgraph, considering the edges node by node—Fig. 3b; while with CEPS, only the nodes must be considered, and no edges at all. In respect to the number of nodes to be considered, with CEPS this number decreases linearly with the number of nodes in the budget; for \(b = 1\), the problem is similar to the manual inspection of the graph, which demands the consideration of all the \(N\) nodes in \(G\). For \(b = N - |Q|\), the problem requires the inspection of only \(|Q|\) nodes—possibly with \(|Q| \ll N\); that is, one must only determine the source nodes that feed the algorithm—Fig. 3c, proceeding interactively to the user’s demand—Fig. 3d. In other words, GMine brings interaction to the broadly studied problem of graph summarization, combining it to hierarchical graph visualization.

### 4.1 CEPS Overview

Given \(Q\) query nodes on a graph, how do we summarize the connectivity relationship among these nodes? The CEPS technique proposes to represent such relationship with a connection subgraph. Such subgraph corresponds to the graph nodes that are center-piece and have direct or indirect connections to all, or most, of the nodes of interest. Formally, given \(Q\) query nodes in a graph \(G = (V, E)\) \((G'\) as a subgraph in a Graph-Tree), find the subset of nodes \(CP \subseteq V'\) that will determine an induced subgraph \(CP \subseteq G'\) with budget \(b\) (maximum \(CP\) size in number of nodes) having strong connections to all \(Q\) query nodes. Following, we will use the symbology presented in Table 1.

![Fig. 3. CEPS visual summarization. (a) A complete graph problem—100 nodes and 4,950 edges. (b) Inspection of the edges of a single node. (c) First summarization with \(Q = 4\) source query nodes and a budget of \(b = 50\) nodes. (d) Further summarization with \(Q = 4\) and \(b = 16\).](image)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G')</td>
<td>the subgraph of a given LeafSuperNode</td>
</tr>
<tr>
<td>(N)</td>
<td>total number of nodes in graph (G')</td>
</tr>
<tr>
<td>(Q)</td>
<td>number of source query graph nodes</td>
</tr>
<tr>
<td>(Q)</td>
<td>set of query graph nodes ((i = 1,\ldots,Q))</td>
</tr>
<tr>
<td>(Q)</td>
<td>(N)-by-1 unit query vector: all zeros except one at row (qi)</td>
</tr>
<tr>
<td>(CP)</td>
<td>the induced center-piece subgraph</td>
</tr>
</tbody>
</table>
A natural way to measure the validity of a subgraph $CP$ is to measure the goodness of the graph nodes it contains: the more "good"/important nodes (with respect to the source queries) it contains, the better $CP$ is. Let us first define the goodness score for nodes. For a given graph node $j$, we have two types of goodness score:

- Let $r(i,j)$ be the goodness score of a given graph node $j$ with respect to the query graph node $q_i$.
- Let $r(Q,j)$ be the goodness score of a given graph node $j$ w.r.t. the query set $Q$.

It follows that the goodness criterion for a $CP$ can be defined as:

$$g(CP) = \sum_{j \in \text{nodes}(CP)} r(Q,j).$$

For this definition, there are two problems to achieve the center-piece subgraph: 1) how to define a reasonable goodness score $r(Q,j)$ for a given graph node $j$; 2) how to quickly find a connection subgraph maximizing $g(CP)$.

### 4.2 Goodness Score Calculation

The concepts for goodness score calculation are:

- Let $r_{i,j}$ be the steady-state probability that a particle will find itself at node $j$, when it does random walk with restart from a query node $q_i$.
- Let $r(Q,j)$ be the meeting probability, that is, the steady-state probability that ALL $Q$ particles, doing RWR from the query nodes of $Q$, will all find themselves at node $j$ in the steady state.

First, we want to compute the goodness score $r(i,j)$ of a single graph node $j$, for a single query node $q_i$. To do so, we use random walk with restart from query node $q_i$. Suppose a random particle starts from node $q_i$, the particle iteratively transmits to its neighborhood with a probability that is proportional to the edge weight between them. Also, at each step, it has a probability $1-c$ to return to node $q_i$. In this conception, $r(i,j)$ is defined as the steady-state probability $r_{i,j}$ that the particle will finally be at node $q_i$:

$$r(i,j) \triangleq r_{i,j}.$$  

Formally, if we put all the $r_{i,j}$ probabilities into matrix form $R = [r_{i,j}]$, then

$$R^T = cR^T \tilde{G}^r + (1-c)E,$$  

where $E = [e_i]$, for $i = 1, \ldots, Q$ is a $N$-by-$Q$ matrix, $c$ is the fly-out probability, and $\tilde{G}^r$ is the (column-) normalized adjacency matrix for graph $G'$. The problem of determining $R^T$ can be solved in many ways—we choose the iteration method, iterating (7) until convergence.

Once $R^T$ is ready, we want to combine the individual scores together to measure the importance for each graph node $j$ w.r.t. the whole query set $Q$. The most common query scenario might be “given $Q$ query nodes, find the subgraph $CP$ whose nodes are important/good w.r.t. ALL query nodes.” In this case, $r(Q,j)$ should be high if and only if there is a high probability that all particles will finally meet at node $j$. This probability is given by

$$r(Q,j) \triangleq r(Q,j; Q) = \prod_{i=1}^Q r(i,j).$$

The goodness score $r(Q,j)$ of a given graph node $j$ w.r.t. the query set $Q$ is the first step in order to calculate the induced center-piece subgraph $CP$. The next step is the “EXTRACT” algorithm.

### 4.3 The “EXTRACT” Algorithm

The “EXTRACT” algorithm takes as input the graph $G'$, the importance/goodness score $r(Q,j)$ on all nodes, and the budget $b$, and produces as output a small, undirected graph $CP$. The basic idea is as follows: 1) instead of trying to find an optimal subgraph maximizing $g(CP)$ directly, we decompose it, finding key paths incrementally; 2) by sorting the graph nodes in order, we can quickly find the key paths by dynamic programming in the acyclic graph.

Before presenting the algorithm, we require the following definitions:

Definition 13. A graph node $u$ is called specified downhill from node $v$ w.r.t. source $q_i$ ($v \rightarrow u_i$) if $r(i,v) > r(i,u)$.

Definition 14. A specified prefix path $P(i,u)$ is any downhill path that starts from source $q_i$ and ends at node $u$; that is, $P(i,u) = (u_0, u_1, \ldots, u_n)$ where $u_0 = q_i, u_n = u$, and $u_j \rightarrow u_{j+1}$, for every $j$.

Definition 15. The extracted goodness is the total goodness score of the nodes within the subgraph $CP$: $CF(CP) = \sum_{j \in CP} r(Q,j)$.

Definition 16. We define an extracted matrix as the matrix whose $(i,u)$ element, $C_{i,u}$, corresponds to the extracted goodness score from a source graph node $q_i$ to node $u$ along the prefix path $P(i,u)$ such that:

1. $P(i,u)$ has exactly $s$ nodes not in the present output graph $CP$.
2. $P(i,u)$ extracts the highest goodness score among all such paths that start from $q_i$ and end at $u$.

In order to discover a new path between the source $q_i$ and a destination node $pd$, we arrange the nodes in descending order of $r(i,j)(j = 1, \ldots, n)$: $\{u_1 = q_i, u_2, u_3, \ldots, pd = u_n\}$. Note that all nodes with smaller $r(i,j)$ than $r(i, pd)$ are ignored. Then, we fill the extracted matrix $C$ in topological order so that when we compute $C_{i,(t,u)}$, we have already computed $C_{i,(t,v)}$ for all $v \rightarrow u$. On the other hand, as the subgraph is growing, a new path may include nodes that are already in the output subgraph. Our algorithm will favor such paths. The complete algorithm to discover a single path from source node $q_i$ and the destination node $pd$ is given in Algorithm 2. Based on the previous preparations, the EXTRACT algorithm is given in Algorithm 3.

The EXTRACT algorithm joins all the formalism presented in this section, the goal is to systematically compute the Center-Piece Subgraph that best summarizes a graph of interest. In Section 6, we present experiments attesting its accuracy and in Section 7 we demonstrate it.

### 5 Graph Tree Performance

Now, we present performance tests for calculating the SuperNode Connectivity (SNC) (Section 3.4.1) and the
Graph Nodes Connectivity (Section 3.4.2). We demonstrate that its performance surpasses that of classic adjacency lists and of relational databases.

**Algorithm 2: Single Key Path Discovery (from node \( i \) to node \( pd \)).**

Let \( Q \) be the set of query nodes;
Let \( len \) be the maximum allowable path length;
Let \( S \) be a set of nodes \( \{u_1, u_2, u_3, \ldots, u_k = u_n\} \), where \( u_k \rightarrow u_{k+1} \) for \( k = 1, \ldots, n-1 \).

for \( j \leftarrow [1, \ldots, n] \) do

for \( s \leftarrow [2, \ldots, len] \) do

if \( v \) is already in the output subgraph then

| \( s' = s \); |

else

\( s' = s - 1 \)
end
Let \( C_s(i, v) = \max_{u \rightarrow i, v} (C_{s'}(i, u) + r(Q, v)) \)
end
Result: The path maximizing \( C_s(i, pd)/s \), where \( s \neq 0 \)

**Algorithm 3: The EXTRACT Algorithm.**

Initialize output graph \( CP \) as an empty graph;
Let \( len \) be the maximum allowable path length;
while \( CP \) is not big enough (i.e., within the budget) do

Pick up destination node \( pd \):
\( pd = \max_{q \in CP} r(Q, j) \);
for each source node \( q \) do

Use Algorithm 2 to discover a key path \( P(q, pd) \);
Add \( P(q, pd) \) to \( CP \);
/*Duplicate path nodes are detected and merged when paths are added to \( CP^* \)*/
end
Result: The final \( CP \)

**5.1 Complexity Analysis**

Considering a k-way partitioned Graph-Tree with \( tn \) nodes (consisting of \( sn \) SuperNodes and \( lsn \) LeafSuperNodes), the height of the tree is given by \( h = \lceil \log_{(tn(k-1) + 1)} \rceil \)—root level is 1—and the number of SuperEdges at level \( l \) is given by \( se(l, k) = (k!/(2(k-2)!)) \). In the configuration of a complete Graph-Tree, \( sn = \sum_{i=1}^{k-1} k^{i-1} \) SuperNodes; \( lsn = k^{h-1} \) LeafSuperNodes; let \( p = |V|/lsn \) be the number of graph nodes per subgraph, \( d = |E|/|V| \) be the average degree of a graph node, and \( r \) be the expected ratio of external edges per graph node, \( 1/d \leq r < 1 \) for \( d > 1 \). Also, let \( f \) be the expected number of edges in a SuperEdge \( r \), where \( Level(r) \) corresponds to the level of the SuperNodes that define \( r \), more specifically, \( f(\text{Level}(r)) = \frac{|E|}{\sum_{i=\text{Level}(r)}^{\text{Level}(r)+1} f(i)} \) if \( \text{Level}(r) = 1 \) and \( f(\text{Level}(r)) = \frac{f(\text{Level}(r)-1)}{\text{Level}(r)} \) if \( \text{Level}(r) > 1 \).

With these parameters, the complexity time for SuperNodes Connectivity, \( SNC(r, GNC) \), is determined by the following factors:

1. time to search for the first common parent, \( \tau_{P_{\tau_1}} \), of \( \tau_P \) and \( \tau_{GNC} \);
2. time to search for the pair of siblings \( (\tau_{\tau_1}, \tau_{\tau_2}) \) beneath \( \tau_P \) in the path to \( \tau_{\tau_1} \) and \( \tau_{\tau_2} \);
3. time to search for the SuperEdge \( (\tau_{\tau_1}, \tau_{\tau_2}) \), and
4. time to perform the verification of which of the edges of \( SuperEdge(\tau_{\tau_1}, \tau_{\tau_2}) \) pertain to the set of possible edges in between \( \tau_P \) and \( \tau_{GNC} \).

The time complexity comes from \( (3*h) + (k) + (2*f*r) \), where \( k \) and \( r \) are constants of the underlying graph, and \( h \) is logarithmic; thus, the complexity is \( O(f) \), where \( f \) is the expected number of edges in a SuperEdge, is a very small fraction of the number of edges \( |E| \).

The Graph Nodes Connectivity, \( GNC(v) \), is given by the time to trace the path from \( v \) to the root; at each level up to the root, it takes the hash time to verify if \( v \) is still an open node and, in each of the elements of the set of \( k-1 \) SuperEdges at a given level, it takes the hash time to track the edges that have \( v \) as an endpoint. Thus, the time complexity comes from \( (h)*((c)+(c+k)) = h + c + k \), where \( k \) is a constant, \( c \) refers to the hash time assumed to be constant, and \( h \) is logarithmic. Then, the chief term is \( h \) and the complexity is logarithmic \( O(h) \) for \( GNC \).

**5.2 Memory Consumption**

Since the Graph-Tree keeps leaf nodes on disk, it provides significant memory gains compared to the adjacency list. This gains depends on factor \( r \), the expected ratio of external edges per graph node; the lower the value of \( r \) the higher are the memory gains because more edges will be on disk and not on memory. In Fig. 4, we present a comparative plot of the memory load for both the Graph-Tree and the adjacency list for a not favorable value of \( r = 0.6 \).

**5.3 Experiments Setting**

We use synthetic graphs with varying number of nodes and average degree of edge. We used graphs with 5K, 10K, 50K, 100K, 500K, and 1M nodes with average edge degree of 3, 12, and 20 edges per graph node; a total of 18 graphs whose number of edges ranges between 15K and 20M edges. We recursively break the graphs at up to five levels and five partitions per level, depending on
the experiment, ranging from 2 to $5^{5 - 1} = 3,125$ partitions. We perform the experiments in a personal computer with a 3 GHz processor, 4 MB L1 cache, 4 GB 500 MHz memory, and a 5,400 rpm 500 GB disk device. The entire experiment (data, code, software, performance measures, and details) is available at http://www.cs.cmu.edu/junio.

The goal is to observe the complexity cost using the wall-clock time necessary to calculate SNC and GNC. The SNC cost is chiefly determined by the expected number of edges ($f$) between the SuperNodes involved in the computation; so we vary this number from 500 to 80K edges. The GNC cost is chiefly determined by the tree height ($h$) where a graph node lies; we use up to five levels from trees that represent small to large scale graphs. We perform all the experiments with the Graph-Tree and the adjacency list; and the first 12 experiments only with the DB2 commodity database.

The Graph-Tree was implemented following Section 3 definitions so that besides a SuperGraph it also provides SNC and GNC functionalities. The adjacency list implementation was made on top of the GraphGarden graph library, under custody of researcher Jure Leskovec (http://www.cs.cmu.edu/jure/). The graph nodes in the list are labeled according to the graph partitioning that they belong to. For maximum performance, the adjacency list uses hash mapping so that the retrieval of a given graph node is done in hash time. We also configured a relational database for the experiment. Its schema defines relations among graph nodes and SuperNodes allowing hierarchical management and SuperNodes' coverage computation. The database uses indexes for optimized searches and redundant information to reduce disk accesses.

5.4 Performance on SNC Computation

The experiments confirmed the analytical expectations for the three different methodologies. The commodity database performance, despite its optimization, declines due to the nested SQL queries necessary for the SNC computation, what implies in random disk accesses. The database performance was one order of magnitude worse than the other two techniques. In turn, the adjacency list performance showed to be linear with the number of nodes and edges, reaching a reasonable performance at the cost of massive memory consumption. The Graph-Tree, on the other hand, is less sensible to these factors, having its performance determined by the size of the answer—that is, the number of edges found in between two arbitrary SuperNodes, a fraction of the graph size (see the analytical calculus of $f$ in Section 5.1).

We note that the different natures of these two techniques ask for specific testing configurations. In Fig. 5a, the parameters of interest are the number of nodes and edges; there we can verify how the adjacency list is more affected by the size of the graph than the Graph-Tree. In Fig. 5b, the parameter of interest is $f$, calculated for several variations of the 18 experimental graphs partitioned according to different levels and numbers of partitions per level. Along with Fig. 5b, Figs. 5c and 5d are intended to elucidate how the measures in Fig. 5b were performed; Fig. 5c shows that the number of graph nodes ranged from 5K to 1 M; Fig. 5d shows that the number of graph edges ranged from 15K to 20 M. Figs. 5a, 5b, and 5c have the same number of points and the same parameter of interest, what makes it possible to join them and see what the performance in seconds of Fig. 5b corresponds to in terms of graph size and, also, to verify empirically that the SCN complexity cost is linear with factor $f$.

The comparison of the methods, in absolute numbers (seconds) was favorable to the Graph-Tree as demonstrated in Fig. 5a. Analytically, speaking the Graph-Tree is favored by two facts; first, the number of external edges only rises to a fraction of the number of graph nodes. Second, even if the graph size increases, a proper partitioning scheme can make the number of external edges grow slower than the growth of the graph size.

5.5 Performance on GNC Computation

For GNC, our first observation is that the performance of the database was almost two orders of magnitude worse than the other two methods; its performance degrades heavily with the increase in the number of graph nodes and edges. The weak performance of the commodity database, once more, is due to the nested queries over the large volumes of information. It is explained by the inadequacy of the relational data model in calculating the GNC, which involves data crossing and tracking of the groups and subgroups to which the graph nodes pertain.

Again here, as we see in Fig. 6a, the adjacency list performance goes with the graph size, having a reasonable performance. Actually, its performance is slightly better than the Graph-Tree for small edge degrees at the expense of larger memory demands. The strong point of the Graph-Tree is that although it is influenced by the graph size, as analytically predicted, its performance is not directly determined by this factor, but by the height ($h$) at which a given graph node of interest lies on—a logarithmically increasing factor.

Just as for the SNC analysis, the different natures of the techniques ask for specific testing configurations. While Fig. 6a is ruled by the number of graph nodes and edges,
Figs. 6b, 6c, and 6d are linked by the same number of points and by the same parameter of interest $h$. The joint of these three figures demonstrate the logarithmic characteristic of the Graph-Tree in numbers; while the curve in Fig.6b range from 0.001 second to nearly 3.5 second, Figs. 6c and 6d show that the average data used during the time experiment ranged from 40K to 540K nodes and from 100K to 8 M edges. We note that average was used because it is not feasible to calculate all the possible hierarchical partitionings given by the combinations of number of levels $h$ and number of partitions per level for each of the 18 graphs, therefore we have uniformly chosen random possibilities and combined their results with average; nevertheless all the possible graph sizes were used.

The GNC computational cost of the Graph-Tree grants a natural scalability potential that is not dictated by the graph size—this is a demand for today’s applications. By using a tree-like graph storage that supports GNC computation, it becomes possible to use all the classical graph algorithms without having the entire graph on memory, providing large scale possibilities.

6 CEPS ACCURACY

In this section, we evaluate the accuracy of the CEPS solution, rather than comparing it to other orthogonal approaches. We are interested in evaluating whether its algorithm captures the most relevant subgraph, given a desired budget size.

The goodness score of an induced subgraph is measured through a simple question: “how much importance is captured by the graph nodes that comprehend an induced subgraph $CP$?” We refer to this measure as the “importance node ratio”, or $I_{Ratio}$. Given a query set $Q$ of nodes, a subgraph $G'$ and a connection subgraph $CP$, the $I_{Ratio}$ refers to the coefficient between the goodness score w.r.t. the induced connection subgraph $CP$ and the goodness score w.r.t. the entire subgraph $G'$. This computation assumes, as discussed in Section 4.2, that the goodness score used by

CEPS is accurate on its goal to measure the goodness of a graph. $I_{Ratio}$ is computed as follows:

$$I_{Ratio} = \frac{\sum_{j \in CP} r(Q, j)}{\sum_{j \in G} r(Q, j)}.$$  

We use the $I_{Ratio}$ to evaluate the quality of CEPS. In our experiments, we apply the CEPS algorithm to the leaf communities of the DBLP data set, each community containing around 500 nodes. Fig. 7 shows the average $I_{Ratio}$ versus size of subgraph (budget); the curves indicate the different query set sizes of our experiments. One can see that a relatively small connection subgraph (with 20 to 30 nodes) can capture most of the important nodes (accounting for $>80\%$ of the total importance). This result shows that the CEPS algorithm sticks to the essence of the original graph as much as possible, while considering the budget size limit.

7 PROOF OF CONCEPT: GMine VISUAL ENVIRONMENT

Here, we introduce the GMine system that, using the Graph-Tree structure, materializes SuperGraphs for visual inspection. Due to space limitations, it is not possible to show all the features of the system, so we have made it available at http://www.cs.cmu.edu/~junio. The data set we use in this paper define authorship graphs deriving from publication data; each graph node represents an author and each edge denotes a coauthoring relationship.

DBLP data set. Here, we present the functionalities of GMine over a larger data set. We use the Digital Bibliography & Library Project (DBLP), a database of Computer Science publications. DBLP defines an authorship graph with 315,688 nodes (authors) and 1,659,853 edges (coauthorings). We use GMine to automatically create a recursive partitioning of DBLP according to the k-way partitioning (METIS). The partitioning has five hierarchy levels, each with five partitions. The data set, thus, is broken into 5$^{k-1}$, or 625, communities with an average of nearly 500 nodes per community. For this data set, such partitioning generates communities anchored on highly collaborative authors and, roughly, on similar research themes.
7.1 Visualization and Interaction

Fig. 8 presents a navigation sequence over DBLP. In Fig. 8a, it is possible to see the five first-level partitions. By observing the SuperNodes connectivity (SuperEdges), it is possible to see that there are three first-level communities highly connected one to each other, and that each of them also has their five subcommunities highly interconnected. The other two first-level communities are relatively isolated, just similarly to their inner subcommunities. It is possible to conclude that the three first-level highly connected communities hold long term collaborating authors, while the other two—s03 and s04—hold less productive casual authors who seldom interact with each other, or with authors from other communities.

In Fig. 8a, we highlight the abstraction-control of GMine (arrow below the figure), which allows to set the control to one of three abstraction entities: the individual graph nodes, the subgraphs at the leaves, or the SuperNodes of the SuperGraph. Fig. 8b focuses on community s04 and also shows (arrow at the left) the levels-selector control of GMine, which permits the navigation through the levels of the hierarchy. In Figs. 8c and 8d, we go deeper into SuperNode s04, focusing on community s043 and, further, on community s0433. Fig. 8d also shows that a leaf community of SuperNode s0433 was loaded from disk (see arrow) under request of the user. In Fig. 8e, community s04333 is then presented with details about the nodes and edges of the correspondent subgraph. At this point, we have reached the deepest level of the SuperGraph. The detailed annotations on community s04333 characterize its parent community s04, which contains mostly isolated nodes at the surroundings, and a few small subgraphs at the center. In Fig. 8f, we focus on one of the subgraphs, which embodies three authors M. Güzelkaya, Eksin, and F. Gürleyen. With the aid of the Graph Node Calculus (Section 3.4.2), we could retrieve their connections to the rest of the graph. We verified that none of them has additional coauthorings and, thus, their subgraph corresponds to their unique publication, dated from 2001.

GMine also supports label search via hashing from the graph nodes to the SuperNodes of the Graph-Tree. In Fig. 9a, we perform a label search for prominent graph analysis researcher Peter Eades; GMine takes us to the correspondent community indicated by the arrow. This subgraph, presented in Fig. 9b, has around 500 nodes cluttered in a limited space. At this point we can apply the CEPS summarization to concentrate on a group of the most interesting graph nodes. As input, we pick authors Peter Eades, Ioannis G. Tollis and Giuseppe Di Battista, defining budget size of 40 as the limit.
for the induced subgraph. Fig. 9c presents the final configuration, in which each graph node is connected to every other by a path smaller or equal 3. The induced graph delineates a collaboration network where the query authors are cornerstone. Interestingly, the subgraph reveals two center-piece authors, Roberto Tamassia and Giuseppe Liotta, as central connections for the summarization subgraph. The entire subgraph presents one of the most remarkable graph research communities in the literature. This is only the main community for author Peter Eades; by calculating the Graph Node Connectivity, we verified that he has other 29 coauthors from other partitions (communities) in this snapshot of DBLP.

8 CONCLUSIONS

We presented GMine, a system for large graphs visual analysis. The framework that supports GMine can process large graphs with hundreds of thousands of nodes using hierarchical graph partitioning and interactive summarization. Contributions include scalability via an innovative formalization for graph hierarchies aimed at graph processing and representation, an innovative connection subgraph extraction algorithm, and a proof-of-concept presentation of large graphs.

As future research, we foresee the Graph-Tree purely designed for disk access, probably having its design oriented to SuperEdges; algorithms over the Graph-Tree for large graphs computation, benefiting from its plenary representation with GNC and SNC; the advancement of the SuperGraph abstraction for dealing with SuperNodes as if they were sole graph nodes, with specific properties reflecting their coverage; and the use of the GMine framework along with state-of-the-art layout techniques both for graphs and graph hierarchies, this last application in demand for systematic user evaluation.

ACKNOWLEDGMENTS

This work was partly supported by Microsoft Research, FAPESP (São Paulo State Research Foundation), CAPES (Brazilian Committee for Graduate Studies), CNPq (Brazilian National Research Foundation), and the US National Science Foundation (NSF) under Grants IIS-0209107, SENSOR-0329549 and IIS-0534205. This work was also partly supported by the Pennsylvania Infrastructure Technology Alliance (PITA) and by donations from Intel, NTT, and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, or other funding parties.

REFERENCES

Jose F. Rodrigues Jr. received the PhD degree from the University of São Paulo, Brazil, part of which was carried out at Carnegie Mellon University in 2007. He is a professor at the University of São Paulo, Brazil. He is a regular reviewer of major conferences in his field having contributed to publications in IEEE and ACM journals and conferences. His topics of research include data analysis, content-based data retrieval and visualization. He is a member of the IEEE.

Hanghang Tong received the PhD degree from the School of Computer Science, Carnegie Mellon University in 2009. He is a researcher at IBM, Hawthorne, New York. He has two conference “best paper” awards and three patents, having also presented several invited talks and tutorials at major conferences. His research interests include large-scale graph mining, anomaly detection, and influence propagation. He is a member of the IEEE.

Jia-Yu Pan received the PhD degree from Carnegie Mellon University, and has received three best paper awards. He is a software engineer at Google, Inc., California, working on anomaly detection and its applications. His research interests include anomaly detection, data mining, web services, and cloud computing.

Agma J.M. Traina is a professor at the University of São Paulo having advised so far 32 graduate students, with more than 100 publications in major journals and conferences. Her research interests include multidimensional indexing methods, information visualization, retrieval by content, image processing, and mining. She is a senior member of the IEEE and the IEEE Computer Society.

Caetano Traina Jr. is a professor at the University of São Paulo, being an active researcher in his field with more than 200 referred papers, several awards and a large history of supervising graduate and undergraduate students. His research interests include database design, indexing methods, similarity queries, and data mining. He is a senior member of the IEEE and the IEEE Computer Society.

Christos Faloutsos is a professor at Carnegie Mellon University. He has received the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), and 17 “best paper” awards. He has published more than 200 refereed articles, and 11 book chapters. His research interests include data mining for graphs and streams, database performance, and indexing for multimedia data. He is a member of the IEEE and the IEEE Computer Society.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.
Visualização de grafos em larga escala
StructMatrix: large-scale visualization of graphs by means of structure detection and dense matrices

Hugo Gualdron, Robson Cordeiro, Jose F Rodrigues Jr

University of Sao Paulo – ICMC, CS Department
Av Trab Sao-carlense, 400, Sao Carlos, SP, Brazil - 13566-590
{gualdron, robson, junio}@icmc.usp.br

Abstract—Given a large-scale graph with millions of nodes and edges, how to reveal macro patterns of interest, like cliques, bi-partite cores, stars, and chains? Furthermore, how to visualize such patterns altogether getting insights from the graph to support wise decision-making? Although there are many algorithmic and visual techniques to analyze graphs, none of the existing approaches is able to present the structural information of graphs at large-scale. Hence, this paper describes StructMatrix, a methodology aimed at high-scalable visual inspection of graph structures with the goal of revealing macro patterns of interest. StructMatrix combines algorithmic structure detection and adjacency matrix visualization to present cardinality, distribution, and relationship features of the structures found in a given graph. We performed experiments in real, large-scale graphs with up to one million nodes and millions of edges. StructMatrix revealed that graphs of high relevance (e.g., Web, Wikipedia and DBLP) have characterizations that reflect the nature of their corresponding domains; our findings have not been seen in the literature so far. We expect that our technique will bring deeper insights into large graph mining, leveraging their use for decision making.

Keywords—graph mining, fast processing of large-scale graphs, graph sense making, large graph visualization

I. INTRODUCTION

Large-scale graphs refer to graphs generated by contemporary applications in which users or entities distributed along large geographical areas – even the entire planet – create massive amounts of information; a few examples of those are social networks, recommendation networks, road nets, e-commerce, computer networks, client-product logs, and many others. Common to such graphs is the fact that they are made of recurrent simple structures (cliques, bi-partite cores, stars, and chains) that follow macro behaviors of cardinality, distribution, and relationship. Each of these three features depends on the specific domain of the graph; therefore, each of them characterizes the way a given graph is understood.

While some features of large graphs are detected by algorithms that produce hundreds of tabular data, these features can be better noticed with the aid of visual representations. In fact, some of these features, given their large cardinality, are intelligible, in a timely manner, exclusively with visualization. Considering this approach, we propose StructMatrix, a methodology that combines a highly scalable algorithm for structure detection with a dense matrix visualization. With StructMatrix, we introduce the following contributions:

1) Methodology: we introduce innovative graph processing and visualization techniques to detect macro features of very large graphs;
2) Scalability: we show how to visually inspect graphs with magnitudes far bigger than those of previous works;
3) Analysis: we analyze relevant graph domains, characterizing them according to the cardinality, distribution, and relationship of their structures.

The rest of the paper presents related works in Section II, the proposed methodology in Section III, experimentation in Section IV, and conclusions in Section V. Table I lists the symbols used in our notation.

II. RELATED WORKS

A. Large graph visualization

There are many works about graph visualization, however, the vast majority of them is not suited for large-scale. Techniques that are based on node-link drawings cannot, at all, cope with the needs of just a few thousand edges that would not fit in the display space. Edge bundling [1] techniques are also limited since they do not scale to millions of nodes and also because they are able to present only the main connection pathways in the graph, disregarding potentially useful details. Other large-scale techniques are visual in a different sense; they present plots of calculated features of the graph instead of depicting their structural information. This is the case of Apolo [2], Pegasus [3], and OddBall [4]. There are also techniques [5] that rely on sampling to gain scalability, but this approach assumes that parts of the graph will be absent; parts that are of potential interest.

Adjacency matrices in contrast to Node-Link diagrams are the most recommended techniques for fine inspection of graphs in scalable manner [6]; this is because they can represent an edge for each pixel in the display. However, even with one edge per pixel, one can visualize roughly a few million edges. Works Matrix Zoom[7] and ZAME[8] extend the one-edge-per-pixel approach by merging nodes and edges through clustering algorithms, creating an adjacency matrix where each position represents a set of edges.
on a hierarchical aggregation. The main challenge of using clustering techniques is to find an aggregation algorithm that produces a hierarchy that is meaningful to the user. There are also matrix visualization layouts as MatLink [9] and NodeTrix [10] combining Node-Link and adjacency representations to increase readability and scalability, but those approaches are not enough to visualize large-scale graphs.

Net-Ray [11] is another technique working at large scale; it plots the original adjacency matrix of one large graph in the much smaller display space using a simple projection: the original matrix is scaled down by means of straight proportion. This approach causes many edges to be mapped to one same pixel; this is used to generate a heat map that informs the user of how many edges are in a certain position of the dense matrix.

In this work, we extend the approach of adjacency matrices, as proposed by Net-Ray, improving its scalability and also its ability to represent data. In our methodology, we introduce two main improvements: (1) our adjacency matrix is not based on the classic node-to-node representation; we first condense the graph as a collection of smaller structures, defining a structure-to-structure representation that enhances scalability as more information is represented and less compression of the adjacency matrix is necessary; and (2) our projection is not a static image but rather an interactive plotting from which different resolutions can be extracted, including the adjacency matrix with no overlapping – of course, considering only parts of the matrix that fit in the display.

B. Structure detection

The principle of StructMatrix is that graphs are made of simple structures that appear recurrently in any graph domain. These structures include cliques, bipartite cores, stars, and chains that we want to identify. Therefore, a given network can be represented in an upper level of abstraction; instead of nodes, we use sets of nodes and edges that correspond to substructures. The motivation here is that analysts cannot grasp intelligible meaning out of huge network structures; meanwhile, a few simple substructures are easily understandable and often meaningful. Moreover, analyzing the distribution of substructures, instead of the distribution of single nodes, might reveal macro aspects of a given network.

Partitioning (shattering) algorithms

StructMatrix, hence, depends on a partitioning (shattering) algorithm to work. Many algorithms can solve this problem, like Cross-associations [12], Eigenspokes [13], and METIS [14], and VoG [15]. We verified that VoG overcomes the others in detecting simple recurrent structures considering a limited well-known set.

VoG relies on the technique introduced by graph compression algorithm Slash-Burn by Kang and Faloutsos [16]. The idea of Slash-Burn is that, in contrast to random graphs or lattices, the degree distribution of real-world networks obeys to power laws; in such graphs, a few nodes have a very high degree, while the majority of the nodes have low degree. Kang and Faloutsos also demonstrated that large networks are easily shattered by an ordered “removal” of the hub nodes. In fact, after each removal, a small set of disconnected components (satellites) appear, while the majority of the nodes still belong to the giant connected component. That is, the disconnected components were connected to the network only by the hub that was removed and, by progressively removing the hubs, the entire graph is scanned part by part. Interestingly, the small components that appear determine a partitioning of the network that is more coherent than cut-based approaches [17]. The technique works for any power-law graph without domain-specific knowledge or specific ordering of the nodes.

For the sake of completeness and performance, we designed a new algorithm that, following the Slash-Burn technique, extends algorithm VoG with parallelism, optimizations, and an extended vocabulary of structures, as detailed in Section III-B. Our results demonstrated better performance while considering a larger set of structures.

III. PROPOSED METHOD: STRUCTMATRIX

As we mentioned before, StructMatrix draws an adjacency matrix in which each line/column is a structure, not a single node; besides that, it uses a projection-based technique to “squeeze” the edges of the graph in the available display space, together with a heat mapping to inform the user of how big are the structures of the graph. In the following, we formally present the technique.

A. Overview of the graph condensation approach

For this work, we use a vocabulary of structures that extends those of former works; it considers seven well-known structures – see Figure 1 – found in the graph mining literature: false stars (fs), stars (st), chains (ch), near and full cliques (nc, fc), near and full bi-partite cores (nb, fb). Shortly, we define the vocabulary of structures as $\psi = \{fs, st, ch, nc, fc, nb, fb\}$.

![Figure 1: The vocabulary of graph structures considered in our methodology. From (a) to (g), illustrative examples of the patterns that we consider; we process variations on the number of nodes and edges of such patterns.](image-url)
Table I: Description of the major symbols used in this work.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(V,E)$</td>
<td>graph with V vertices and E edges</td>
</tr>
<tr>
<td>$S, S_ε$</td>
<td>structure-set</td>
</tr>
<tr>
<td>$n,</td>
<td>S</td>
</tr>
<tr>
<td>$M, m_{x,y}$</td>
<td>StructMatrix</td>
</tr>
<tr>
<td>$fc, nc$</td>
<td>full and near clique resp.</td>
</tr>
<tr>
<td>$fb, nb$</td>
<td>full and near bipartite core resp.</td>
</tr>
<tr>
<td>$st, fs, ch$</td>
<td>star, false star and chain resp.</td>
</tr>
<tr>
<td>$ψ$</td>
<td>vocabulary (set) of structures</td>
</tr>
<tr>
<td>$D(s_i, s_j)$</td>
<td>Number of edges between structure instances $s_i$ and $s_j$</td>
</tr>
</tbody>
</table>

False stars are structures similar to stars (a central node surrounded by satellites), but whose satellites have edges to other nodes, indicating that the star may be only a substructure of a bigger structure—see Figure 1. A near-clique or $ε$-near clique is a structure with $1 - ε$ ($0 < ε < 1$) percent of the edges of a similar full clique would have; the same holds for near bipartite cores. In our case, we are considering $ε = 0.2$ so that a structure is considered near clique or near bipartite core, if it has at least 80 percent of the edges of the corresponding full structure.

The rationale behind the set of structures $ψ$ is that (a) cliques correspond to strongly connected sets of individuals in which everyone is related to everyone else; cliques indicate communities, closed groups, or mutual-collaboration societies, for instance. (b) Chains correspond to sequences of phenomena/events like those of “spread the word”, according to which one individual passes his experience/feeling/impression/contact with someone else, and so on, and so forth; chains indicate special paths, viral behavior, or hierarchical processes. (c) Bipartite cores correspond to sets of individuals with specific features, but with complementary interaction; bipartite cores indicate the relationship between professors and students, customers and products, clients and servers, to name a few. And, (d) stars correspond to special individuals highly connected to many others; stars indicate hub behavior, authoritative sites, intersecting paths, and many other patterns.

Considering these motivations, our algorithm condenses the graph in a dense adjacency matrix. To do so, it produces a set with the instances of structures in $ψ$ that were found in the graph; this set of instances contains the same information as that of the original graph but with vertices and edges grouped as structures. Beyond that, the algorithm detects the edges in between the structures, so that it becomes possible to build a condensed adjacency matrix that informs which structure is connected to each other structure.

**B. StructMatrix algorithm**

As mentioned earlier, our algorithm is based on a high-degree ordered removal of hub nodes from the graph; the goal is to accomplish an efficient shattering of the graph, as introduced in Section II-B. As we describe in Algorithm 1, our process relies on a queue, $Φ$, which contains the unprocessed connected components (initially the whole graph), and a set $Γ$ that contains the discovered structures. In line 4, we explore the fact that the problem is straight parallelizable by triggering threads that will process each connected component in queue $Φ$. In the process, we proceed with the ordered removal of hubs—see line 5, which produces a new set of connected components. With each connected component, we proceed by detecting a structure instance in line 7, or else, pushing it for processing in line 10. The detection of structures and the identification of their respective types occur according to Algorithm 2, which uses edge arithmetic to characterize each kind of structure.
C. Adjacency Matrix Layout

A graph $G = (V, E)$ with $V$ vertices and $E$ edges can be expressed as a set of structural instances $S = \{s_0, s_2, \ldots, s_{|S|-1}\}$, where $s_i$ is a subgraph of $G$ that is categorized – see Figure 1 and Table I – according to the function $\text{type}(s) : S \to \psi$. To create the adjacency matrix of structures, first we identify the set $S$ of structures in the graph and categorize each one. Following, we define $n = |S|$ to refer to the cardinality of $S$.

As depicted in Figure 2, each type of structure defines a partition in the matrix, both horizontally and vertically, determining subregions in the visualization matrix. In this matrix, a given structure instance corresponds to a horizontal and to a vertical line (w.r.t. the subregions) in which each pixel represents the presence of edges (one or more) between this structure and the others in the matrix. Therefore, the matrix is symmetric and supports the representation of details. The sum of nodes of two given connected structures. We use a color map in which the smaller number of nodes is indicated with bluish colors and the bigger number of nodes is indicated with reddish colors. In addition, we use the same information as used for color encoding to determine the order of plotting: first we plot the edges of the smaller structures (according to the number of nodes), and then the edges of the bigger structures. This procedure assures that the hotter edges will be over the cooler ones, and that the interesting (bigger) structures will be spotted easier.

At this point the elements $m_{i,j}$ of a StructMatrix $M_{n \times n}$, $0 < i < (n - 1)$ and $0 < j < (n - 1)$ are given by:

$$m_{i,j} = \begin{cases} 1, & \text{if } D(s_i, s_j) > 0; \\ 0, & \text{otherwise.} \end{cases}$$

where $D : S \times S \to \mathbb{N}$ is a function that returns the number of edges between two given structure instances. For quick reference, please refer to Table I.

In this work, we focus on large-scale graphs whose corresponding adjacency matrices do not fit in the display. This problem is lessened when we plot the structures-structures matrix, instead of the nodes-nodes matrix. However, due to the magnitude of the graphs, the problem persists. We treat this issue with a density-based visualization for each subregion formed by two types of structures $(\psi_i, \psi_j), \psi_i \in \psi$ and $\psi_j \in \psi$ – for example, $(fs, fs), (fs, st), \ldots$, and so on. In each subregion, we map each point of the original matrix according to a straight proportion. We map the lower, left boundary point $(x_{min}, y_{min})$ to the center of the lower, left boundary pixel, and the upper, right boundary point $(x_{max}, y_{max})$ to the center of the upper, right boundary pixel.

The remaining points are mapped as $(x, y) \mapsto (\rho_x, \rho_y)$ for:

$$\rho_x = R(\psi_i, \psi_j) + \left(\text{Res}_x - 1\right) \frac{x - x_{min}}{x_{max} - x_{min}} + \frac{1}{2}$$

$$\rho_y = R(\psi_i, \psi_j) + \left(\text{Res}_y - 1\right) \frac{y - y_{min}}{y_{max} - y_{min}} + \frac{1}{2}$$

where $R : \psi \times \psi \to \mathbb{N}$ is a function that returns the offset (left boundary) in pixels of the region $(\psi_i, \psi_j)$ and $\text{Res}_x, \text{Res}_y$ are the target resolutions. The more resolution, the more details are presented, these parameters allow for interactive grasping of details.

Each set of edges connecting two given structures is then mapped to the respective subregion of the visualization where the structures’ types cross. Inside each structure subregion we add an extra information by ordering the structure instances according to the number of edges that they have to other structures; that is, by $\sum_{i=0}^{\psi-1} D(s, s_i)$.

Therefore, the structures with the largest number of edges to other structures appear first – more at the bottom left, less at the top right, of each subregion as explained in Figure 2.

In the visualization, each horizontal/vertical line (w.r.t. the subregions) corresponds to a few hundred or thousand structure instances; and each pixel corresponds to a few hundred or thousand edges. We deal with that by not plotting the matrix as a static image, but as a dynamic plot that adapts to the available space; hence, it is possible to select specific areas of the matrix and see more details of the edges. It is possible to regain details until reaching parts the original plot, when all the edges are visible.

We plot one last information using color to express the sum of nodes of two given connected structures. We use a color map in which the smaller number of nodes is indicated with bluish colors and the bigger number of nodes is indicated with reddish colors. In addition, we use the same information as used for color encoding to determine the order of plotting: first we plot the edges of the smaller structures (according to the number of nodes), and then the edges of the bigger structures. This procedure assures that the hotter edges will be over the cooler ones, and that the interesting (bigger) structures will be spotted easier. At this point the elements $m_{i,j}$ of a StructMatrix $M_{n \times n}$, $0 < i < (n - 1)$ and $0 < j < (n - 1)$ are given by:

$$m_{i,j} = \begin{cases} C(\text{NNodes}(s_i) + \text{NNodes}(s_j)), & \text{if } D(s_i, s_j) > 0; \\ 0, & \text{otherwise.} \end{cases}$$

where $\text{NNodes} : S \to \mathbb{N}$ is a function that returns the number of nodes of a given structure instance; and
$C : \mathbb{N} \rightarrow [0.0, 1.0]$ is a function that returns a continuous value between 0.0 (cool blue for smaller structures) and 1.0 (hot red for bigger structures) according to the sum of the number of nodes in the two connected structures. In our visualization, we map the function $C$ to a log scale and then we apply a linear color scale to the data.

IV. EXPERIMENTS

Table II describes the graphs we use in the experiments.

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Edges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>1,366,099</td>
<td>5,716,654</td>
<td>Collaboration network</td>
</tr>
<tr>
<td>Roads of PA</td>
<td>1,088,092</td>
<td>1,541,898</td>
<td>Road net of Pennsylvania</td>
</tr>
<tr>
<td>Roads of CA</td>
<td>1,965,206</td>
<td>2,766,607</td>
<td>Road net of California</td>
</tr>
<tr>
<td>Roads of TX</td>
<td>1,379,917</td>
<td>1,921,660</td>
<td>Road net of Texas</td>
</tr>
<tr>
<td>WWW-barabasi</td>
<td>325,729</td>
<td>1,090,108</td>
<td>WWW in nd.edu</td>
</tr>
<tr>
<td>Epinions</td>
<td>75,879</td>
<td>405,740</td>
<td>Who-trusts-whom network</td>
</tr>
<tr>
<td>cit-HepPh</td>
<td>34,546</td>
<td>420,877</td>
<td>Co-citation network</td>
</tr>
<tr>
<td>Wiki-vote</td>
<td>7,115</td>
<td>100,762</td>
<td>Wikipedia votes</td>
</tr>
</tbody>
</table>

Table II: Description of the graphs used in our experiments.

A. Graph condensations

Table III shows the condensation results of the structure detection algorithm over each dataset, already considering the extended vocabulary and structures with minimum size of 5 nodes – less than 5 nodes could prevent to tell apart the structure types. The columns of the table indicate the percentage of each structure identified by the algorithm. For all the datasets, the false star was the most common structure; the second most common structure was the star, and then the chain, especially observed in the road networks. The improvement of the visual scalability of StructMatrix, compared to former work Net-Ray, is as big as the amount of information that is “saved” when a graph is modeled as a structure-to-structure adjacency matrix, instead of a node-to-node matrix.

B. Scalability

In order to test the processing scalability of StructMatrix, we used a breadth-first search over the DBLP dataset to induce subgraphs of different sizes – we created graphs ranging from 50K edges up to 1.000K edges. For the scalability experiment, we used a contemporary commercial desktop (Intel i7 with 8 GB RAM). We compared the performance between VoG and StructMatrix to detect simple recurrent structures from a limited well-known set. Figure 5 shows that StructMatrix and VoG are near-linear on the number of edges of the input graph, however StructMatrix overcomes VoG for all the graph sizes.

C. WWW and Wikipedia

In Figures 3 and 4, one can see the results of StructMatrix for graphs WWW-barabasi (325,729 nodes and 1,090,108 edges) and Wikipedia-vote (7,115 nodes and 100,762 edges) condensed as described in Table III. For graph WWW-barabasi, Figure 3a shows the StructMatrix with linear color encoding, and Figure 3b shows the StructMatrix with (a) Normal scale. (b) Log scale.

Figure 3: StructMatrix in the WWW-barabasi graph with colors displaying the sum of the sizes of two connected structures; in the graph, stars refer to websites with links to other websites.

(c) Normal scale. (d) Log scale.

Figure 4: StructMatrix in the Wikipedia-vote graph with values displaying the sum of the sizes of two connected structures; in this graph, stars refer to users who got/gave votes from/to other users.
Table III: Structures found in the datasets considering a minimum size of 5 nodes.

<table>
<thead>
<tr>
<th></th>
<th>fs (%)</th>
<th>st (%)</th>
<th>ch (%)</th>
<th>nc (%)</th>
<th>fc (%)</th>
<th>nb (%)</th>
<th>fb (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>122,983(76%)</td>
<td>7,585(5%)</td>
<td>3,096(2%)</td>
<td>2,656(2%)</td>
<td>24,551(15%)</td>
<td>14(&lt;1%)</td>
<td>-</td>
</tr>
<tr>
<td>WWW-barabasi</td>
<td>4,957(32%)</td>
<td>8,146(52%)</td>
<td>851(5%)</td>
<td>541(3%)</td>
<td>283(2%)</td>
<td>556(4%)</td>
<td>318(2%)</td>
</tr>
<tr>
<td>cit-HepPh</td>
<td>11,449(79%)</td>
<td>1,948(13%)</td>
<td>840(6%)</td>
<td>120(1%)</td>
<td>44(4%)</td>
<td>35(&lt;1%)</td>
<td>43(&lt;1%)</td>
</tr>
<tr>
<td>Wikipedia-vote</td>
<td>1,112(65%)</td>
<td>564(33%)</td>
<td>29(2%)</td>
<td>-</td>
<td>-</td>
<td>1(&lt;1%)</td>
<td>-</td>
</tr>
<tr>
<td>Epinions</td>
<td>4,518(52%)</td>
<td>2,725(31%)</td>
<td>1,247(14%)</td>
<td>28(%)</td>
<td>21(%)</td>
<td>150(2%)</td>
<td>3(&lt;1%)</td>
</tr>
<tr>
<td>Roadnet PA</td>
<td>11,825(23%)</td>
<td>22,934(45%)</td>
<td>13,748(27%)</td>
<td>-</td>
<td>-</td>
<td>2,668(5%)</td>
<td>-</td>
</tr>
<tr>
<td>Roadnet CA</td>
<td>24,193(27%)</td>
<td>34,781(39%)</td>
<td>26,236(29%)</td>
<td>-</td>
<td>-</td>
<td>3,763(4%)</td>
<td>-</td>
</tr>
<tr>
<td>Roadnet TX</td>
<td>15,595(25%)</td>
<td>27,094(43%)</td>
<td>17,457(28%)</td>
<td>-</td>
<td>-</td>
<td>2,468(4%)</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 5: Scalability of the StructMatrix and VoG techniques; although VoG is near-linear to the graph edges, StructMatrix overcomes VoG for all the graph sizes.

Figure 3b refer to sites with multiple pages and many out-links – bigger sites are reddish, more connected sites to the left. The visualization is able to indicate the big stars (sites) that are well-connected to other sites (reddish lines), and also the big sites that demand more connectivity – reddish isolated pixels. The chains indicate site-to-site paths of possibly related semantics, an occurrence not so rare for the WWW domain. There is also a set of reasonably small, interconnected sites that connect only with each other and not with the others – these sites determine blank lines in the visualization and their sizes are noticeable in dark blue at the bottom-left corner of the star-to-star subregion. Such sites should be considered as outliers because, although strongly connected, they limit their connectivity to a specific set of sites.

While the Wikipedia graph is mainly composed of stars, just like the WWW graph, the Wikipedia graph is quite different. Its structures are more interconnected defining a highly populated matrix. That means that users (contributors) who got many votes to be elected as administrators in Wikipedia, also voted in many other users. The sizes of the structures, indicated by color, reveal the most voted users, positioned at the bottom-left corner – the color pretty much corresponds to the results of the elections: of the 2,794 users, only 1,235 users had enough votes to be elected administrators (nearly 50% of the reddish area of the matrix). There are also a few chains, most of them connected to stars (users), especially the most voted ones – it becomes evident that the most voted users also voted on the most voted users. This is possibly because, in Wikipedia, the most active contributors are aware of each other.

D. Road networks

On the road networks, if we consider the stars segment (“st”), each structure corresponds to a city (the intersecting center of the star); therefore, the horizontal/vertical lines of pixels correspond to the more important cities that act as hubs in the road system. Its StructMatrix visualization – Figure 7 – showed an interesting pattern for all the three road datasets: in the figure, one can see that the relationships between the road structures is more probable in structures with similar connectivity. This fact is observable in the curves (diagonal lines of pixels) that occur in the visualization – remember that the structures are first ordered by type into segments, and then by their connectivity (more
Another interesting fact is the presence of some structures heavily connected to nearly all the other structures; these structures define horizontal lines of pixels in the visualization and, due to symmetry, they also define vertical lines of pixels. The same patterns were observed for roadnets from California, Texas, and Pennsylvania. According to the visualizations, roads are characterized by three patterns:

1) cities that connect to most of the other cities acting as interconnecting centers in the road structure; these cities are of different importance and occur in small number – around 6 for each state that we studied;

2) there is a hierarchical structure dictated by the connectivity (importance) of the cities; in this hierarchy, the connections tend to occur between cities with similar connectivity; one consequence of this fact is that going from one city to some other city may require one to first “ascend” to a more connected city; actually, for this domain, the lines of pixels in the visualization correspond to paths between cities, passing through other cities – the bigger the inclination of the line, the shorter the path (the diagonal is the longest path);

3) road connections that are out of the hierarchical pattern – the ones that do not pertain to any line of pixels; such connections refer to special roads that, possibly, were built on specific demands, possibly not obeying to the general guidelines for road construction.

From these visualizations and patterns, we notice that the StructMatrix visualization is a quick way (seconds) to represent the structure of graphs on the order of million-nodes (intersections) and million-edges (roads). For the specific domain of roads, the visualization spots the more important cities, the hierarchy structure, outlier roads that should be inspected closer, and even, the adequacy of the roads’ inter connectivity. This last issue, for example, may indicate where there should be more roads so as to reduce the pathway between cities.

**E. DBLP**

In the StructMatrix of the DBLP co-authoring graph – see Figure 6a – it is possible to see a huge number of false stars. This fact reflects the nature of DBLP, in which works are done by advisors who orient multiple students along time; these students in turn connect to other students defining new stars and so on. A minority of authors, as seen in the matrix, concerns authors whose students do not interact with other students defining stars properly said. The presence of full cliques (fc) is of great interest; sets of authors that have co-authorship with every other author. Full cliques are expected in the specific domain of DBLP because every paper defines a full clique among its authors – this is not true for all clique structures, but for most of them.

In Figure 6b, we can see the full clique-to-full clique region in more details and with some highlights indicated by arrows. The Figure highlights some notorious cliques:

\( k_1 \) refers to the publication with title “A 130.7mm 2-layer 32Gb ReRAM memory device in 24nm technology” with 47 authors; \( k_2 \) refers to paper “PRE-EARTHQUAKES, an FP7 project for integrating observations and knowledge on earthquake precursors: Preliminary results and strategy” with 45 authors; and \( k_3 \) refers to paper “The Biomolecular Interaction Network Database and related tools 2005 update” with 75 authors. These specific structures were noticed due to their colors, which indicate large sizes. Structures \( k_1 \) and \( k_3 \), although large, are mostly isolated since they do not connect to other structures; \( k_2 \), on the other hand, defines a line of pixels (vertical and horizontal) of similarly colored dots, indicating that it has connections to other cliques.

**V. CONCLUSIONS**

We focused on the problem of visualizing graphs so big that their adjacency matrices demand much more pixels than what is available in regular displays. We advocate that these graphs deserve macro analysis; that is, analysis that reveal the behavior of thousands of nodes altogether, and not of specific nodes, as that would not make sense for such magnitudes. In this sense, we provide a visualization methodology that benefits from a graph analytical technique. Our contributions are:

- **Visualization technique:** we introduce a processing and visualization methodology that puts together algo-
Figure 7: StructMatrix with colors in log scale indicating the size of the structures interconnected in the road networks of Pennsylvania (PA), California (CA) and Texas(TX). Again, stars appear as the major structure type; in this case they correspond to cities or to major intersections.

Our approach can provide interesting insights on real-life graphs of several domains answering to the demand that has emerged in the last years. By converting the graph’s properties into a visual plot, one can quickly see details that algorithmic approaches either would not detect, or that would be hidden in thousand-lines tabular data.

ACKNOWLEDGMENTS

We thank Prof. Christos Faloutsos and Dr. Danai Koutra, from Carnegie Mellon University, for their valuable collaboration. Furthermore, this work received support from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq-444985/2014-0), Fundacao de Amparo a Pesquisa do Estado de Sao Paulo (FAPESP-2011/13724-1, 2013/03906-0, 2014/07879-0, 2014/21483-2), and Coordenacao de Aperfeiçoamento de Pessoal de Nivel Superior (Capes).

REFERENCES


Capítulo G. Visualização de grafos em larga escala
Visualização de Bancos de Dados Relacionais por meio de grafos
Graph-based Relational Data Visualization

Daniel Mário de Lima∗, José Fernando Rodrigues Jr.† and Agma Juci Machado Traina‡
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, Brazil
Email: {∗danielm, †junio, ‡agmal}@icmc.usp.br

Abstract—Relational databases are rigid-structured data sources characterized by complex relationships among a set of relations (tables). Making sense of such relationships is a challenging problem because users must consider multiple relations, understand their ensemble of integrity constraints, interpret dozens of attributes, and draw complex SQL queries for each desired data exploration. In this scenario, we introduce a twofold methodology; we use a hierarchical graph representation to efficiently model the database relationships and, on top of it, we designed a visualization technique for rapidly relational exploration. Our results demonstrate that the exploration of databases is deeply simplified as the user is able to visually browse the data with little or no knowledge about its structure, dismissing the need for complex SQL queries. We believe our findings will bring a novel paradigm in what concerns relational data comprehension.

Keywords—relational databases; hierarchical visualization; visual analytics;

I. INTRODUCTION

Over the last decades, a huge amount of information has been generated, making large databases common in several kinds of applications. Examples of this growth are found in industry-generated data, where information from clients, products, and transactions of multiple types are stored in relational manner. In these databases, the entities are described by attributes and refer to each other in relationships that define a strong structural cohesion. Relational Database Management Systems (RDBMS’s) are the ultimate solution for such structured data; they provide intelligent storage, and powerful querying capabilities in applications that range from commerce to education. And, despite the fact that non-relational solutions have popped-up in the market – such as NoSQL databases, RDBMS’s still answer for the greatest market-share [1], [2]. But, although RDBMS’s are wonderfully engineered for data storage, they are not adequate for visual analytics – this is the point we tackle in this study.

In this context, storing and retrieving data is only part of the problem when the goal is to obtain useful information. Reasoning about voluminous and complex data can be very difficult, a shortcoming that demands analytical processes in order to find patterns, uncommon arrangements and relationships, and other kinds of hidden knowledge that may aid in decision support. Such processes are commonly performed in exploratory fashion according to which the analyst does not know a priori what to look for. In these circumstances, a rich and responsive visual environment can provide preferred results, notably for structured data.

A straightforward approach to investigate structured data is to use graph representations with node-link visualizations, according to which nodes and edges correspond, respectively, to entities and relationships of the Entity-Relationship (ER) database model [3]. Within these considerations, here we experiment with a graph representation based on hierarchical partitioning, a technique that improves the scalability of graph-based visualizations. And so, our proposal uses the ER structure in order to generate an initial graph that is hierarchically partitioned according to the entities, attributes, and values found in the database. This hierarchically-partitioned graph, then, gives rise to a multiple-level visualization comprised of nodes, groups of nodes, edges, and summarizations over which interactive querying and aggregation take place.

In this study, given a relational database, we are interested in answering the following questions:

- How are the data (entity instances) distributed over the relations of the database?
- How are the entities on the database related to each other?
- How do the several attributes of the database influence the relationships of the entities?
- How do we quickly and intuitively browse the relational data, considering its complex structure?

We answer these questions by using hierarchical graph partitionings created from both the structure and the data found in the database to be analyzed. Over that, we define a novel visual/interactive scheme instantiated in a fully operational prototype. Our contribution makes the exploration of the relationships between data entities intuitive and computationally fast, even if considering very large databases. According to our technique, the database structure can be browsed through exploration paths in which the user can visualize entities and their relationships without explicitly defining queries. Our study relies on the SuperGraph concept and the Graph-Tree structure [4], and we derive the GMine system [5], originally designed for sole graph analysis, to the realm of visual analysis of relational data; hence, our
new systematization is named RMine system. Coming next, Section II reviews the related literature, followed by our proposed methodology in Section III, where we will define: (a) a relational-based hierarchical partitioning for databases; (b) how this partitioning is represented in a Graph-Treestructure together with associated algorithms; and (c) the visualization environment for this data structure. In Section IV, our experiments will show the main aspects of our approach and how it simplifies the data manipulation process. Then, in Section V we finish with a brief discussion of the main achievements, outlining ideas for future improvements.

II. RELATED WORK

Although there are many studies on database visualization, these studies do not explore the structural information of the data, but rather they concentrate on table representation in cube-like schemes. We introduce a novel perspective, one that permits to explore how multiple tables relate to each other.

One of the most referenced studies that aim at visually exploring databases is Polaris [6], [7]. It follows the well-known data cube approach, which is broadly used in a number of decision support systems in businesses and organizations, mainly in the form of OLAP (on-line analytical processing) services [8]. The system provides an interface to develop and interact with visual specifications: a visual specification is the assignment of the attributes of a table for each of the axes of a data cube, along with the required definitions for: selection of records, data transformations, aggregations, partitionings, sortings, and visualization properties. Polaris’ data-cube is organized so that each cell presents the visualization (scatterplots, or barcharts) of specific data ranges over selected attributes at different granularities. Different from our study, Polaris is not designed for the inspection of relational structure data.

In another study [9], Stolte et al. propose Zoom Graphs, a general multiscale visualization of hierarchically structured data. Their study defines a formal notation to design zoom-graph visualizations; the notation uses four possible patterns used to describe the core structure of the most common designs. Although this method can model complex schemata by using multiple hierarchies, it restricts the user’s interaction to follow a single exploration path through a previously chosen hierarchy.

On a different line, Maniatis et al. [10] employ the Table Lens technique [11] over data cube visualizations. They adopt the Cube Presentation Model to split the presentation components of the logical data layer, allowing the user to explore sections of a fact table by choosing the desired values of the attributes being presented. Similarly, Techapichetchavanich and Datta [12] introduce the Hierarchical Dynamic Dimensional Visualization (HDDV) to explore hierarchically structured data from data cubes. Their approach maps the cube dimensions to the levels of an exploration tree whose visualization exposes the exploration path. In the tree, each level is a bar stick with dividing marks for splitting either attribute ranges or nominal labels of a given dimension. This method allows the user to visually build cube queries and quickly switch to a different hierarchical path, but it is still able to show only a single path at a time.

Mansmann and Scholl [13] improve the visualization of multiple hierarchies with the Enhanced Decomposition Tree, which combines the Cube Presentation Model [10] with different space-filling visualization techniques. In their scheme, the exploration tree is allowed to have sub-trees comprising different attributes and being visualized in parallel.

Wang et al. [14] present a client-server visualization system named Zoom Tree. Their system allows a schema-based data cube navigation similar to the schema of Mansmann and Scholl [13], with the difference that dimensions with too many values are presented according to hierarchies organized by ranges of values. Selected portions of the data are visualized in a table layout inspired by Polaris [6], and the user’s navigation (zooming into the table data) is stored in the Zoom Tree, thus forming a navigation history.

ASK-GraphView et al. [15] offers a hierarchical graph structure which closely resembles the SuperGraph concept [4]. Their approach partitions the graph in antichains whose associated views can be processed in the main memory, then reducing the depth by retracting combs and reducing the fan-out by separating siblings under new subtrees. With this structure, they apply a recursive clustering pipeline to find subgraph components and create the hierarchical visualization. Differently from their approach, the proposed method employs the database structural information to partition the graph and build the hierarchy.

A recurrent concern of these studies is the scalability regarding both response time and visual cluttering, fundamental demands for interactive systems. The data cube visual interface, as used in these studies, is well-established, providing a promising environment for exploratory analysis. However, these studies are tied to quantitative analysis, demanding the analyst to know the right attributes and operations in order to compose the visualizations; such schemes provide little or no support for exploring relationships between entities.

Besides, all these previous studies are transactions-oriented, focusing on the datacube metaphor; differently, our study does not focus on transactions, but on the multiple relationships that emerge from the structure of operational databases.

III. THE PROPOSED APPROACH

A. SuperGraph overview

The method we propose is based on the concept of SuperGraphs, a formalization that abstracts a hierarchically partitioned graph. A SuperGraph is recursively comprised
of nodes, SuperNodes (groups of nodes), edges, and SuperEdges (groups of edges), and is defined as follows:

**Definition 1: [SuperGraph]** Given a finite undirected graph \( G = (V, E) \), with neither loops nor parallel edges, a SuperGraph is defined as \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} \) is a set of SuperNodes \( \mathcal{F} \), and \( \mathcal{E} \) is a set of SuperEdges \( \mathcal{F} \).

**Definition 2: [SuperNode]** A SuperNode \( \mathcal{V} \) is recursively defined as a set \( \mathcal{V} \) of SuperNodes or graph nodes (if it is a leaf), plus a set \( \mathcal{E} \) of SuperEdges \( \mathcal{F} \). As follows:

\[
\mathcal{V} = \{v_0, v_1, ..., v_{(\mathcal{V}_{n-1})}\}, \\
\mathcal{E} = \{v_i: (v_i, v_j) \in \mathcal{V}\} \\
\]  

(1)

**Definition 3: [SuperEdges]** A SuperEdge represents all the edges \( (u, v) \in E \) that connect graph nodes from a SuperNode \( \mathcal{V} \) to graph nodes from SuperNode \( \mathcal{V} \). Formally, the SuperEdge between SuperNodes \( \mathcal{V} \) and \( \mathcal{V} \) is defined as follows:

\[
SuperEdge(\mathcal{V}, \mathcal{V}) = \mathcal{E} = \{e = (u, v) \mid (u, v) \in E, \\
u \text{ is enclosed in } v_i \text{ and } v \text{ is enclosed in } v_j\} \\
\]  

(2)

**Definition 4: [Weight of a SuperEdge]** The weight of a SuperEdge is the sum of the weights of its edges.

Figure 1 exemplifies the SuperGraph abstraction. Figure 1(a) shows graph \( G \), defined as \( G = (V, E) = \{(1, 7), (1, 8), (2, 7), (3, 4), (3, 7), (4, 5), (4, 6), (7, 8)\} \).

From graph \( G \), it is possible to conceive the hierarchical partitioning presented as SuperNode \( \mathcal{G} \) in Figure 1(b). This partitioning is composed of SuperNodes \( \mathcal{V} \) through \( \mathcal{V} \) and corresponding SuperEdges:

\[
\mathcal{V} = \{\mathcal{V}, \mathcal{V}\}, \mathcal{E} = \{\mathcal{E}_{11} = \{(3, 7)\}\} \\
\mathcal{V} = \{\mathcal{V}, \mathcal{V}\}, \mathcal{E} = \{\mathcal{E}_{12} = \{(1, 7), (1, 8), (2, 7)\}\} \\
\mathcal{V} = \{\mathcal{V}, \mathcal{V}\}, \mathcal{E} = \{\mathcal{E}_{22} = \{(4, 5), (4, 6)\}\} \\
\mathcal{V} = \{\mathcal{V}, \mathcal{V}\}, \mathcal{E} = \{\mathcal{E}_{33} = \{(7, 8)\}\} \\
\mathcal{V} = \{\mathcal{V}, \mathcal{V}\}, \mathcal{E} = \{\mathcal{E}_{44} = \{(3, 4)\}\} \\
\mathcal{V} = \{\mathcal{V}, \mathcal{V}\}, \mathcal{E} = \{\mathcal{E}_{55} = \{(5, 6)\}\} \\
\]

Figure 1(c), in turn, presents the corresponding Graph-Tree structure, which reflects the hierarchical partitioning of the SuperGraph. In the figure, one can see that the Graph-Tree is designed so that leaf SuperNodes are selectively loaded from disk. The main feature of the Graph-Tree is its ability to dynamically determine the edges that interconnect either nodes or SuperNodes. This feature implies that:

1) given a node, one can determine all the edges that connect to this node without having to check all the partitions and levels of the graph hierarchy;

2) given any two SuperNodes, one can determine all the edges that connect these two groups of nodes.

These two features are the key to the present study as they allow for the dynamic inspection of the structural data in relational database. In order to provide these features, the Graph-Tree uses specific algorithms whose details are out of the scope of this study.

**B. Relational-based hierarchical partitioning**

In previous studies, the Graph-Tree data structure was used to process and to visually interact with graphs that were automatically partitioned. The problem with this application is that the number \( h \) of levels in the hierarchy is determinant in interpreting the graph hierarchical partitioning; however, there are no algorithms to automatically determine these values from a given graph. To solve this problem, in this study, we define the number of hierarchy levels as the number of attributes of interest in a given database relation. As it will be explained further on, this defines a semantically-rich hierarchy organized according to the values of the attributes found in the data.

Together with this approach, we use the information given by the relationships between the different entities of the database to instantiate graph-like data. Therefore, we were able to produce hierarchically-partitioned graphs that incorporate the information of entire databases considering the semantics given by their attributes and the structure given by their relationships. This approach is totally different if compared to former database visualization techniques, which are centered on quantitative transactional values, disregarding the important information represented by the structure of the database.

**First level**

Following our line of thought, the first level of the hierarchy is determined by entities that define many-to-many relationships. For example, consider the simple database schema shown in Figure 2 – in this schema, entity Person has a many-to-many relationship to entity Publication. Following the SuperGraph representation, the first level of the hierarchy will have one partition (or SuperNode) for each corresponding entity – Person and Publication, in this case. These SuperNodes are added as children of the root node.

By using this initial partitioning, the correspondent database hierarchical visualization will present the database entities in its first level, as illustrated in Figure 3(a). In this
first level, the edges between Person and Publication defines a SuperEdge composed by all the edges between nodes in these partitions. The first level of the visualization, hence, provides an overview of how the data is structured and how the different entities manifest in the database. In system RMine, this visualization is interactive. Besides visualizing the SuperNodes, it is possible to retrieve further details by inspecting the SuperEdges of the visualization. And so, one can double click a SuperEdge of interest and the underlying Graph-Tree will present which nodes interact with each other.

**Deeper-levels**

Following, the idea is to have the possibility of refining the information of each SuperNode entity of the first level. That is, each SuperNode had to be partitioned into another set of SuperNodes, a level below in the hierarchy. The problem here is how to determine the partitioning in a given level, and how many levels to use. In order to answer these two questions, our method considers the attributes of each entity as the information to guide the lower levels of the partitioning. We proceed considering two actions:

a) the number and meaning of the levels are given by the attributes of each entity, one level per attribute;

b) the number of partitions in a given level is given by the values of that attribute as found in the database.

Action a) implies that the most representative attributes of a given entity must be considered. In the example, entity Person can be represented by age and city, determining two levels below the first level; and entity Publication can be represented by country and year, again two levels below the first level - illustrated in Figure 2. Action b) is a little more complicated; it demands to search the database and verify the distribution of values for each attribute.

There are different kinds of attributes; the main kinds are categorical, nominal, and numeric. Categorical attributes with small numbers of categories, as gender for example, will determine one partition for each distinct category. For a greater number of categories, it is interesting to evenly partition the data following its distribution. For nominal and numeric attributes, the number of partitions can be determined by considering the distribution of the values. Figure 2(b) shows an example with each attribute being partitioned in two groups – attribute age partitions the Person SuperNode in two SuperNodes “age > 50” and “age < 50” and then each SuperNode is again partitioned in two SuperNodes, “city: Paris” and “city: Seoul”. This method
results in the Graph-Tree structure illustrated in Figure 2(c), which can be used in a hierarchical SuperGraph visualization as illustrated in Figure 3. According to such visualization, the user can descend in different paths and levels of the tree to inspect the SuperEdges between the various arrangements of SuperNodes.

Still for nominal and numeric attributes, one can consider a greater number of $k$ partitions; which can be achieved with traditional statistic percentile analysis. Since we are dealing with a visual interactive ensemble, the default value of $k$ is set around the limits of human working memory, which is $7+/-2$ elements according to Miller’s Law [16]. This restriction leads to a number of partitions per level that does not overload the user’s cognition.

As an example, Figure 4 shows the distribution of ages of a real-world database (detailed in Section IV), in which we define $k = 5$ partitions. We obtain quintiles $QU_1 = [18, 41]$ (20 % of the observations), $QU_2 = [42, 49]$ (40 % of the observations), $QU_3 = [50, 55]$ (60 % of the observations), $QU_4 = [56, 63]$ (80 % of the observations) and $QU_5 = [64, 162]$ (100 % of the observations).

Following the percentile statistical approach, we end up with partitions with nearly the same size in number of entity nodes. This is a straightforward procedure intended for automatic analysis; yet, our methodology can also benefit from user-defined parameters (such as a greater $k$) for specific analytical goals.

Design criteria
The design of our methodology was defined to satisfy four features:

1) the structural information of the database becomes represented in the graph hierarchy, an information formerly ignored;
2) the semantic information given by the attributes and values found in the database are kept for further analysis;
3) it becomes possible to deal with very large databases while maintaining its semantic meaning and interpretability;
4) the hierarchical representation adheres to hierarchical visualization techniques, permitting to visually explore the database.

In Section IV, we demonstrate these features together with additional quantitative criteria by means of a broad set of experiments.

C. Database representation and preprocessing
In order to have the database represented as a semantic-rich hierarchical graph, as described in the preceding section, it is necessary to preprocess it based on choices of representative attributes. This step involves selecting the relevant relations (entities) and relationships, as required by the analyst, tracing its set of attributes and their respective distributions.

Once the database is preprocessed, its data is scanned and a Graph-Tree is created following a hierarchical partitioning that conforms to the data properties. In an instantiated Graph-Tree there are three kinds of information:

- the nodes at the leaves, which correspond to the tuples of the relations;
- the SuperNodes along the hierarchy up to the first level, which correspond to sets (groups) of nodes and, recursively, to sets of SuperNodes; at the first level, they correspond to the relations of the database;
- structural information that distributed along the structure permits to calculate the relationships (edges) between any pair node-node, node-SuperNode, or SuperNode-SuperNode.

From the storage point of view, a small portion of the Graph-Tree is kept in the main memory, while a bigger portion is kept on the disk. This organization permits to have the data loaded on demand, saving on processing and memory resources.

D. Hierarchical Graph Visualization

Our method generates semantic-rich SuperGraph for a given database, having as a product, a Graph-Tree that organizes and manages data on demand. The conception of this product, as it has been described, adheres to hierarchical visualization techniques of any kind (as those listed by Schulz [17]), and as a proof-of-concept, we use a canonical node-link visualization as implemented in GMine. Originally, this visualization has been used to visualize graphs in general; for the database domain, we have adapted it so to bring the specificities of relational data to an interactive visual environment.

The visualization we propose is based on the following operations, which can be thought as specialized cases of drill-down and roll-up operations [18]:

- expand SuperNode: digs one level down the hierarchy;
- contract SuperNode: moves one level up in the hierarchy;
- SuperNode-to-SuperNode connectivity: as one expands or contracts a given SuperNode, its correspondent SuperEdges must reflect the operation, that is, subsets/supersets of SuperEdges must be calculated for every pair of SuperNodes;
- hide/show SuperEdge: as the number of SuperEdges varies, the user must be able to hide or show SuperEdges of interest;
- expand SuperEdge: under user’s demand, the set of edges that determine a given SuperEdge is presented in a separate visualization, in which details can be observed;
- filter edges: as a SuperEdge is expanded, the number of edges presented to the user may be way too much for visualization; for this, our scheme supports filtering...
edges based on their weight, that is, the number of times two nodes have interacted in the database.

All these features are integrated in a novel environment for database visualization, as illustrated in the next section. In this environment, once the database data are loaded, the user can verify and refine the information that characterizes each entity, at the same time that he/she can track the relationships within these entities.

IV. EXPERIMENTS

To demonstrate the potential of the proposed approach, this section covers some experiments. Given a database and some exploratory tasks, we illustrate each task together with its corresponding SQL queries and processing times. The intent is to demonstrate that our method substitutes complex and costly SQL operations that, otherwise, would demand time to be written and to be processed. Our method, instead, allows for the same operations satisfying to interaction-time restrictions.

A. Database and test setup

The data used in the following experiments come from the Tycho-USP database. Tycho-USP is a general academic database from the University of São Paulo, containing data about students, professors, and their academic work. The data are collected from various information systems in the university and merged with external data from scientific agencies of the Brazilian government. It is structured in a schema with five main entities: Event (352,400 nodes), Examination (382,890 nodes), Publication (691,083 nodes), Research Supervision (26,237 nodes) and Person (50,779 nodes); with relationships Person-Examination (851,168 edges), Person-Event (247,516 edges), Person-Publication (691,083 edges), Person-Supervision (52,439 edges) and Publication-Event (26,237 edges); a total of 1,503,389 nodes and 1,868,443 edges.

As illustrated in Figure 5, the four entities of Tycho-USP were considered according to their attributes of interest as follows: Person (age, location, gender), Publication (year, country, subject), Event (rank, sphere, type), Supervision (type, progress, advisor_role) and Examination (country, course, institution). The structure of the database was considered according to the relationships linking Person to all the other entities, and Publication to Event. The attributes were used to determine the levels of the hierarchy, and the relationships were used to determine the edges of the underlying graph.

For our experiments, we use parameter $k = 5$. The nominal attributes are divided into the $k$ most frequent classes plus an “others” class with the remaining elements; and numeric attributes are partitioned in $k$ percentiles with approximately the same cardinality. The first task is to build the data structure; to this end, our method receives a configuration set containing the entities and relationships of interest and builds an empty Graph-Tree. Then, the construction procedure writes the nodes and edges in the leaf SuperNodes, and fills up the upper levels with connectivity SuperEdges. This initial step creates a persistent Graph-Tree on disk, which can be loaded in the visualization system later.

All time measurements are averaged wall-clock timings, taken on a personal computer equipped with an AMD Phenom II X4 850 processor, 4 GB of DDR3 main memory, a single 500 GB SATA hard disk and Microsoft Windows 7 Professional x64 operating system.

B. Visual analysis

In this section, we demonstrate how our method can be used for visually inspecting a relational database. We carry the following tasks:

1) Visualize the distribution of the entities and of their relationships: how many people, publications and events are there and how are they arranged?

2) Visualize the relationship between Person and Examination: which countries are preferred by exchange students?

3) Which courses are preferred in Brazil?

When the Graph-Tree is ready, RMine presents the first level under the root, one SuperNode for each database entity. As noted in Figure 6(a), by selecting one of the SuperNodes, RMine calculates and presents the SuperEdges that connect this SuperNode to the other entities. The size of the SuperNodes are proportional to the number of nodes they represent, and the thickness of the SuperEdges are $c \cdot \log(n)$ to the number $n$ of edges they represent and a configurable constant $c$. Therefore, one can intuitively tell which are the largest relations (Publication and Event) and which are the most intense relationships (Person-Examination and Person-Publication), just as addressed by question 1). Figure 6 also presents the correspondent SQL commands that would be necessary to produce the same information as that being visually observed.

The next interaction step is to expand one SuperNode of interest. This action triggers the next level of partitioning, according to the first attribute. Each SuperNode expansion triggers a number of connectivity calculations...
that assign new SuperEdges between the newly exposed children SuperNodes and the remaining SuperNodes in the visualization. In our example, we expand SuperNode Examination according to the country where the examination occurred – see Figure 6(b), which highlights the connections to France, and Figure 6(c), which emphasizes entity Person and the parametric aspect of the corresponding SQL. And so, for the Person-Examination relationship, we can answer question 2) by simply reading the weights of the connectivity SuperEdges – they sum up the number of USP students in all countries, pointing to partitions: Brazil, Spain, United States, France, Portugal, and others.

After expanding more entities, the visualization will look like Figure 7, in which a subset of partitions is presented in deeper levels of the hierarchy. Looking closely at SuperNode Person, now partitioned by age, we see that the automatic partitioning makes each one of the Person by age SuperNodes hold a percentile range with an approximately equal number of objects. The figure shows that roughly 20% of the people in this database are older than 42 years old and younger than 49 years old. Now, considering entity Publication, as partitioned by year, one can see that the ranges of the partitions tend to shorten; since the partitioning followed the percentile approach, it means an increase in number of publications along the time.

At this point, by inspecting SuperNode Examination by country, we can answer question 3). For this task, we expand SuperNode Examination:Country=Brazil – Figure 8(a), so that we can view the most representative courses together with connectivity SuperEdges to Person by age – Figure 8(b). The visualization shows that Business, Architecture, Law, Nursing, and Education are the preferred courses, and for a given selected course, we can inspect how are the preferences by age. For Architecture course, we can see that younger people (18 through 41 years) answer for the smallest fraction of the people (professors and students) related to this specific course.

From another point of view, we might be interested in analyzing the behavior of the younger fraction of people in relation to all the other courses – Figure 9. The produced visualization points out that Law is the course to which younger professionals and students are most connected to. In RMine, we can go deeper in our data inspection...
by retrieving the specific instances that correspond to the observed SuperEdges. For example, if we are interested in knowing details about People:Age=[18-41] and Examination:Country=Brazil:Course=Business, a simple click on its correspondent SuperEdge will bring us the visualization presented in Figure 10. There, in a bipartite graph, one can see the names of the people and the titles of the examinations that are related in the specific context of their correspondent SuperNodes.

This section demonstrated that our method allows for complex aggregation tasks to be performed intuitively in interaction-time constraints. We showed that, not only the user is spared the need to write complex SQL code, but also that the processing requirements are significantly reduced. The cost of these benefits is the preprocessing time that, also, is within acceptable time constraints; especially if we consider the fact that the Graph-Tree structure is persistent on disk.

C. Time measurements

In the previous section, we demonstrated the visual interactive appeal of our method, which lends itself to database exploratory analysis; we also showed how our method prevents users from writing complex SQL queries by means of the Graph-Tree features. Besides these contributions, our method permits costly relational aggregations to be executed in a fraction of the time a relational database would take for the same task. In this section, we demonstrate this feature by comparing the wall-clock timing both for the database execution and for the RMine processing. To this end, we will use 374 different connectivity (SuperNode-to-SuperNode) computations, each one corresponding to a complex SQL aggregation query. For our time measures, we use commodity database PostgreSQL 9.2.1 amd64 with
Consider the connection SuperEdges between Examination by country by course and Person by age. This information can be accomplished interactively or using an SQL query. As an illustration, the connectivity between SuperNode Examination:Country=Brazil:Course=Business and SuperNode Person:Age=[18-41] can be selected with the following SQL query:

```sql
SELECT person.name, examination.title FROM exam
JOIN person ON person.id = exam.person
    AND (person.age BETWEEN 18 AND 41)
JOIN examination
    ON examination.id = exam.examination
    AND (examination.country = 'Brazil')
    AND (examination.course = 'Business');
```

The same holds for the 374 connectivity calculations proposed in this section. The calculations vary, both for the commodity RDBMS and for our method, depending on the intensity of the relationship between the SuperNodes involved in the analysis. Hence, we show that the time we take is significantly smaller than the time an RDBMS would take; note that we are not considering the time a user would take to design the SQL queries, this time is supplanted by the interactive aspect already presented in the former section.

In Figure 11 we show the time for each of the queries considered in the experiment. The figure shows that our method achieves the answer faster than the RDBMS in a great number of cases, mainly in heavier steps. The database caching also plays a clear role, showing slower queries interleaved by faster ones. The figure also shows that the time taken varies depending on the attributes involved in the query; some nominal attributes determine unbalanced partitionings, and thus, unbalanced result sets. In other words, some queries return more data than others. Figure 12 presents the accumulated time for the queries. The figure shows that, for sequences of queries, our method progresses arithmetically better than the RDBMS; this was a need in the design of our method because exploratory interaction asks for long sequences of trial and error steps.

In Table I we list some times taken by our method and by the RDBMS. Column Load corresponds to the time RMine takes to load data from disk; column Conn corresponds to the time an expansion operation (connectivity calculation) takes in RMine – an expansion triggers a set of connectivity calculations from the expanded SuperNode to the other SuperNodes in the scene; and column SQL corresponds to the time the RDBMS takes to do the same expand operations. Still in Table I, we consider rows for the initial load of the preprocessed Tycho-USP Graph-Tree, and rows for expand operations considering entities Person, Examination, Event, and Publication. All times are in seconds. The totalizations in the table demonstrate that all steps of the visual interaction were computed faster in RMine than the corresponding relational queries. In fact, RDBMS’s are not designed for exploratory data inspection, and this is the point we attack.

<table>
<thead>
<tr>
<th>Expand SuperNode</th>
<th>Load (seconds)</th>
<th>Conn (seconds)</th>
<th>SQL (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>0.042</td>
<td>5.955</td>
<td>12.833</td>
</tr>
<tr>
<td>Person:50-55</td>
<td>0.014</td>
<td>1.731</td>
<td>2.752</td>
</tr>
<tr>
<td>Examination</td>
<td>0.276</td>
<td>7.133</td>
<td>9.513</td>
</tr>
<tr>
<td>Examination:U.S.A.</td>
<td>0.000</td>
<td>2.571</td>
<td>2.598</td>
</tr>
<tr>
<td>Event</td>
<td>0.168</td>
<td>4.420</td>
<td>9.253</td>
</tr>
<tr>
<td>Event:Regional</td>
<td>0.003</td>
<td>0.398</td>
<td>0.713</td>
</tr>
<tr>
<td>Publication</td>
<td>0.244</td>
<td>8.584</td>
<td>29.989</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8.321</strong></td>
<td><strong>39.376</strong></td>
<td><strong>67.651</strong></td>
</tr>
</tbody>
</table>
V. Conclusion

We have defined and experimented on a novel approach for analyzing the structure, the data, and the relationships as defined in relational databases. Our solution was based on the Graph-Tree structure and its related algorithms, which provided an efficient way of storing, retrieving, and calculating the relationship information of the database, features that are key to our method. Over the Graph-Tree, we defined a procedure for reading and organizing the database information according to a semantic-rich hierarchical graph partitioning. The Graph-Tree, then, was used as the basis of RMine, an operational prototype for relational visual analysis.

We worked with a visual graph-based approach that demonstrated to be intuitive in respect to visual exploration, and that proved to be efficient in terms of computational cost. The visual exploration spares the analyst of the need to write complex SQL queries, meanwhile the computational cost benefits from the efficient relationship features provided by the Graph-Tree. For future studies, we envision the coupling of analytical features to aid the user in summarizing the meaning of the multiple data presented over the visualization; also, we consider the possibility of having the Graph-Tree to be dynamically altered according to analytical parameters on the fly.

Acknowledgment

This study received support from the following funding agencies: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes).

References


Capítulo H. Visualização de Bancos de Dados Relacionais por meio de grafos
Hierarchical visual filtering
Hierarchical Visual Filtering, pragmatic and epistemic actions for database visualization

Jose F Rodrigues Jr
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
junior@icmc.usp.br

Carlos E Cirilo
Antonio F Prado
Computing Department
Fed. University of São Carlos
São Carlos, SP, Brazil
cirilo@dc.ufscar.br,
prado@dc.ufscar.br

Luciana A M Zaina
Computing Department
Fed. University of São Carlos
Sorocaba, SP, Brazil
lzaina@ufscar.br

ABSTRACT
Visualization techniques of all sorts suffer from visual cluttering, the occlusion of visual information due to the overlap of graphical items; and from excessive complexity in analytical tasks due to multiple parallel perspectives. To cope with these problems, we introduce Hierarchical Visual Filtering, a novel interaction principle based on pragmatic and epistemic actions. Pragmatic actions here mean that the analyst is able to visually select and filter information, determining visual configurations that reveal different perspectives; epistemic actions mean that the analyst can record, annotate, and recall intermediate visualizations created pragmatically. To do so, we use a tree-like organization to keep multiple visualizations of a workspace linked according to the analytical decisions taken by the user. Our goal is to promote an innovative systematization that can augment the potential for database visual inspection, and for visualization systems in general. It is our contention that Hierarchical Visual Filtering can inspire a novel scheme of visualization environments in which space limitations and complexity are treated by means of interactive tasks.

Keywords
Information Visualization; Multiple Views; Visual Data Analysis; Databases; Interactive Filtering; Hierarchical Filtering

1 Introduction
The growth in information production has addressed the problem of accessing the riches of information embedded in large and complex databases. This issue has aggrivated in yearly basis and one of the strategies to deal with it is to use visualization. However, sole visualization techniques are naturally limited in space. Users cannot tell apart items nor distinguish regions of interest when visualizations exceed the available display space, or when data dimensionality prevents effective presentations. Even using well-known interaction mechanisms such as interactive filtering [16], hierarchical parallel coordinates [6], focus+context [1], and link & brush [3], the grasping of interesting facts becomes a memory intensive task. A problem even worse when more than one investigation task is being performed over the same scene; in this situation, it is up to the analyst to remember the appropriate interaction parameters for each line of investigation and to switch in between them. These restrictions in data presentation lead to the need of interface and interaction designs that can aid the management of cognitive load in visual decision-support.

Cognitive aspects
In cognitive science, Kirsh and Maglio [11] identified two kinds of actions performed during decision-support activities. Pragmatic actions are performed to bring one closer to a goal, and epistemic actions are performed to uncover information that is hard to compute mentally. In arithmetic, for example [9], pragmatic actions permit to gradually advance on a problem’s state in order to reach its solution. Epistemic actions correspond to various intermediate results (paper notes), which theoretically could be stored in working memory, but that are recorded externally to reduce cognitive loads. In the realm of Information Visualization (InfoVis) applications, pragmatic actions can be identified as interaction operations to lead the analyst in the task of discovering useful knowledge. Epistemic actions correspond to the recording of intermediate visual presentations in order to assist the analyst in a sequence of interactive steps. More precisely, in what refers to computational aided visualization, Kirsh and Maglio state that epistemic actions correspond to an automatic views management system whose advantages include:

1. reduced memory involved in visual analysis – space complexity;
2. reduced number of steps involved in visual analysis – time complexity;
3. reduced probability of error of visual analysis – reliability.

In a level higher than cognitive science, these advantages have been perceived by Grinstein and Ward [8] who enunciated that limitations in screen resolution and color perception can be solved through multiple linked visualizations. One main approach is to use multiple views enabled with focus+context functionalities to partition and detail intricate visualizations. Indeed, partitioning a dataset is critical to select relevant data and to reduce data for further investigation. Dividing the scene into multiple views help to create smaller subsets easier to be managed and that allow comparative discernment over different perspectives. Chi et al [2] state that a single complex view can be cognitively overwhelming. Multiple views can help the user to “divide and conquer” aiding memory by reducing the amount of data one needs to consider at the same time.
Visualizing relational databases

In the last decade, many works have addressed the issue of visualizing relational databases to formulate new hypotheses, to validate known hypotheses, or to simply present facts more intuitively. These works include multivariate multiple-view systems, and relational database driven systems.

Among the multivariate multiple-view systems, the GGobi system [4], and the Xmdv tool [17] employ linked views as defined by the Link & Brush principle. Our work differs from these former proposals by means of the hierarchical visual filtering principle. For this reason, our work cannot be directly compared to these other proposals, due to its innovative interaction scheme, and to its database orientation.

More specific to our work, in the field of relational-database driven systems, Wang et al. [20] present the ZoomTree, a web-based system that uses a grid layout to present sequential zoom operations carried over a database. Different from our approach, the ZoomTree allows data propagation via selection of dimensions to draw aggregation datacubes; also, the selection of data does not follow free database querying, instead, it is restricted to datacube-refinement operations. A pioneer work on the same topic is system Polaris [18], another grid-based layout whose main feature is its flexibility in defining the visual encoding; different from our work, Polaris does not allow hierarchical exploration, rather it improves on the commercial Pivot Table method. Another related proposal is the work of Mansmann and Scholl [12]; their Decomposition Tree permits the user to visually conduct datacube operations, that is, dimension-oriented operations that are data-driven, rather than analysis-driven. As a last remark, we can state that our work is defined over a combination of visualization with clutter reduction techniques; in this scenario, even if we consider the taxonomic review of clutter-reduction techniques of Ellis and Dix [5], we find that our interaction scheme is a novel contribution.

Visual filtering

Visual filtering (or brushing) over multiple coordinated views [15] is one of the main interaction techniques for visualization; it allows analysts to select the graphical items that are more interesting, gaining focus and details over them. In a recent work, Weaver [22] introduces a scheme that is opposite to ours. Instead of creating multiple views by means of visual filtering, they define a collective filtering based on selections originating from multiple views. Kehrer et al. [10] argue in favor of statistical summarizations (mean, variance, skewness, and kurtosis), each in a dedicated view, as a means for interpreting visually filtered data. Also recent, Turkay et al. [19] create multiple views by simultaneously filtering data and data dimensions assisted by statistical summaries. In another work, Weaver [21] proposes the use of boolean logic to progressively refine the content visually presented, leading to a more robust filtering. Our work differs from all these proposals as it permits an unrestricted number of views that are kept linked by complimentary actions; besides, our proposal is adequate for any kind of visualization technique, including statistical summaries as well.

The rest of the text is organized as follows. In section 2 we present the innovations proposed by our system along with concepts, analyses, and discussion. In section 3 we formally analyze the potential of our interaction principle regarding visual clutter. In section 4 we demonstrate our technique in respect to pragmatic and epistemic actions, just before the final remarks of section 5.

2 Methodology

As a critical setting for database visual analysis, we consider the use of heterogeneous visualizations in a multiple views environment making use of pragmatic and epistemic actions. In order to prove this concept, we developed Visualization Tree (VisTree) [13], a system that enables multiple representations of a dataset relation allowing comparison and operation with a strong feeling of locus of control. VisTree is designed for any kind of visualization technique; specifically for this work, we use classical techniques Parallel Coordinates, Scatter Plots, Table Lens [14], and Fastmap-based projection [7]. Over the VisTree systematization we demonstrate the Hierarchical Visual Filtering principle.

2.1 Hierarchical Visual Filtering

A straightforward way to partition a dataset is to use relational database queries that, over visualization scenes, correspond to interactive filtering, a well-known pragmatic instrument of visualization. Extending this interactive principle, here we propose what we call Hierarchical Visual Filtering, an improvement of interactive filtering that brings epistemic possibilities to visual analysis. Hierarchical Visual Filtering allows for pragmatic actions as the analyst is able to select parts of the visualization scene; and allows for epistemic actions as the analyst is able to record, annotate, and recall intermediate visualizations created over his pragmatic actions. Hierarchical Visual Filtering contributes to visual analysis research in two ways:

- by reducing visual clutter for more scalable analysis;
- by reducing cognitive loads – specially over memory, for a richer analytical experience.

Hierarchical Visual Filtering combines interactive selection and progressive refinement to permit analytical management in a hierarchically-arranged environment. As presented in Figure 1, it is composed of a dataset \(D\), a visualization \(V\), a visualization function \(v\), an interactive filtering function \(f\) and a function \(\Lambda\) that plays just reverse to function \(v\).

In Figure 1, function \(v: D_1 \to V_i\) is parameterized by a pair \((s, g)\) where \(s\) is the spatialization scheme that states how data \(D\) occupy the screen space (projection, graph-like, sequentially, and so on) and \(g\) is the set of graphical marks (dots, lines, curves, icons, and so on) used by visualization \(V_i\). Also in Figure 1, filtering function \(f: V_i \to V_i'\) is parameterized by a pair \((d, e)\) where \(d\) is the set of dimensions of the data and \(e\) is a set of relational select predicates that apply to \(d\) in order to determine the selection of the interactive filtering. Interactive filtering produces an altered configuration \(V_i'\) that is made of a subset of the graphical entities of visualization.
Function $\Lambda : V'_i \rightarrow D_{i+1}$ receives a filtered visualization $V'_i$ and performs the opposite of function $v$ using its same parameters; it returns the data items $D_{i+1}$ that define visualization $V'_i$.

Figure 1 presents the iteration cycle that defines Hierarchical Visual Filtering. According to it, a dataset $D_1$ is used to create a visualization $V_1$ through function $v$. This visualization is interactively filtered via function $f$ to determine a new visualization configuration $V'_1$. Then, the data that is being presented in $V'_1$ is extracted with function $\Lambda$. With the extracted data $D_{i+1}$, a new visualization $V_{i+1}$ can be created. This process is to be repeated iteratively according to the user’s exploratory goals, which determines how many steps are necessary.

In Figure 2, we illustrate the Hierarchical Visual Filtering the way it is performed in VisTree and according to the technique presented in Figure 1. We use two iteration sequences, in the first one the entire dataset (“SELECT * FROM DATA_SET”) $D_1$ is loaded in a new visualization workspace; the interface of this workspace permits to choose one of five available visualization techniques – Figure 2(a). By choosing one of them, visualization function $v$ is triggered to generate visualization $V_1$. Over $V_1$, the user can determine parameters (highlighted in the figure) for interactive function $f$, and a new visualization $V'_1$ is determined. Another user command (Pipeline – Figure 2(b)) triggers function $\Lambda$, which identifies the data $D_2$ being presented in $V'_1$ (“SELECT * FROM DATA_SET WHERE (139 ≤ DIM_2 ≤ 334) AND (73 ≤ DIM_4 ≤ 78)”). The data of $V'_1$ is now sent (pipelined) to a new workspace to create visualizations $V_2$ and then $V'_2$ in the second iteration cycle – Figure 2(c).

Hierarchical Visual Filtering permits the user to define new visualization workspaces, each one carrying a subset of the data that he/she considers worthy for analysis. To do this, the user can observe a given workspace and interactively filter its data by choosing a set of parameters $e$ of relational select predicates. When an interesting visualization configuration comes up, the user can command its data elements to flow to a new workspace. This process can be repeated for the new workspace or for the same workspace. As another workspace is created, the current branch of exploration becomes deeper, as seen in Figure 3(a). If a new workspace is created from the same workspace, a parallel branch of exploration is created, as seen in Figure 3(b). The same operation can be done at any level of the tree, as demonstrated in Figure 3(c). Along this iteration/interaction, the VisTree systematization is responsible for keeping track of the multiple workspaces and to present them in a tree-like structure.

The epistemic aspect of VisTree comes as the system automatically records the relational predicates used during analysis and, also, it comes as the user is able to annotate each workspace so that she/he can remember what the presented data refers to. VisTree complements these features by organizing the workspaces in accordance to how they were created, providing a sense of analytical flow with reduced cognitive load.

2.2 Tree management and interaction

As the user pragmatically creates new workspaces and the tree structure evolves, it is necessary to manage its structure in order to maximize space utilization and to maintain the epistemic potential of the tree representation. Each node of the tree occupies an equal area of the 2D space, therefore it is necessary to partition this space at the same time that the tree representation adjusts to its spatial arrangement.

We developed a recursive algorithm to combine suitable tree presentation with space partitioning of display space. The algorithm uses a tree scheme named $\text{Node}$ – presented in Figure 4(a), which carries pointers to son and sibling nodes. It also carries a pointer to an instance of a visualization object, referenced as $\text{vis}$ pointer. The tree data structure grows as the user triggers interaction events, as illustrated in Figures 4(b) and 4(c). The structure is used during the whole management of the visualization environment.

![Figure 2: Hierarchical Visual Filtering iteration loop. (a) Choice for visualization function $v$; (b) triggering of function $\Lambda$; (c) second iteration cycle.](image)

![Figure 3: VisTree Construction. (a) Sequential exploration of a branch. (b) Parallel exploration over different branches. (c) Iteration at deeper levels of the tree.](image)

![Figure 4: (a) The node structure for managing the VisTree. (b) A tree example and its correspondent data structure. (c) The $\text{vis}$ pointers hold the instances of the visualizations (workspaces) being presented.](image)
In VisTree, the tree is presented in left to right orientation, instead of top-down as usual. To organize nodes this way, we break the node positioning problem in two parts; first, to determine horizontal positioning and, then, to determine vertical positioning. For a given node, horizontal positioning is straightly achieved using the node’s (horizontal) level, which corresponds to how deep the node is located in the tree, as can be seen in Figure 5. Meanwhile, vertical positioning is a little trickier because it demands positioning from the bottom of the display up to its top considering the number of workspaces and of branches at each level. Therefore, tree presentation must start by the leaves, otherwise we may have edge crossings and/or nodes overlapping that could compromise the tree aesthetics.

In Figure 5 we illustrate how the presentation process occurs. The first thing is that, any new node is positioned at the bottom of the screen. In steps 1 through 3 in Figure 5, nodes 5 and 6 illustrate the case for nodes without sons: initial bottom positioning, search for the first node above it (surrounded by circle) and then repositioning. In steps 4 through 8, nodes 7 and 8 illustrate the case for nodes with sons; after the first repositioning, it is necessary to perform a second repositioning according to the child nodes (surrounded by an ellipse). For aesthetic reasons, this second repositioning places the father node in a position where both the sum and the variance of distances for all child nodes are minimized. That is, the father node is positioned in the middle height, which ranges from its highest son (or grandson) to its lowest son (or grandson).

![VisTree node positioning](image.png)

Figure 5: VisTree’s nodes positioning. The first illustration presents the desired presentation. Following, steps 1 through 8 illustrate the positioning of nodes 5 through 8 in order to achieve the desired configuration. For each positioned node, its “first node above” is highlighted with a circle. Child sons that are considered for repositioning are highlighted with ellipses.

### 2.3 Workspaces management and interaction

In the environment of the VisTree, as new epistemic information is recorded in the form of new workspaces, the tree grows and its nodes become smaller in terms of screen space. However, the nodes of the tree have a minimum acceptable size, otherwise, too small visualization scenes could not be observed. For this minimum size, we use 2.5 x 1.5 inches, which nearly corresponds to a two-column article illustration. We assume that the screen resolution will allow us to perform a second repositioning according to the child nodes (surrounded by an ellipse). For aesthetic reasons, this second repositioning places the father node in a position where both the sum and the variance of distances for all child nodes are minimized. That is, the father node is positioned in the middle height, which ranges from its highest son (or grandson) to its lowest son (or grandson).

Our strategy to handle this matter is to provide automatic and manual interaction mechanisms to explore the visualization environment. Hence, when the tree does no longer fit the screen space and a new node is created, we automatically position this last node at the center of the screen. With this mechanism we provide as much context as possible for the new node; at the same time, we induce locus of control, because the last user operation is immediately presented to him/her surrounded by the formerly created workspaces.

But, as we provide endless visualization space, the drawback is that the tree can be much bigger than the screen space and acting epistemically becomes a usability problem. To tackle with this, we designed an interaction scheme enabled with zooming and translation functionalities. These two features aid in the process but, for large trees, they may demand excessive interaction steps in order for the user to grasp the desired information. Therefore, we also designed a novel interaction scheme with two features, named Instant Focus and Instant Context:

- **Instant Context**, Figure 6 (a), works by presenting the entire tree structure no matter how small its nodes become. To do so, the user has to pass the mouse over a desired node and the tree will be visualized with emphasis over that node. It becomes possible to figure out where in the tree structure the node is located. As the user moves the cursor out of the desired node, the application returns to the former spatial arrangement;
- **Instant Focus**, Figure 6(c), allows the user to pop up a separate window from the tree structure; this new window bears the visualization workspace that the user has chosen for focusing. To do so, similarly, the user must pass the mouse cursor over the desired workspace. The focus window can occupy part of the screen space or can be maximized for the entire screen. To return to the VisTree environment, the user has to pass the mouse cursor out of the boundaries of the focus window or to close the window in case it has been maximized.

Instant Focus and Instant Context permits the user to benefit from focus+context interaction over the VisTree multiple windows environment.

![VisTree interaction](image.png)

Figure 6: Instant Context shows where in the tree structure a node of interest is located (a). For large trees, normal presentation may hide context and/or detail (b). Instant Focus presents the details of a given scene by increasing its window size (c).

### 3 Reduction of visual clutter

In this section we formally demonstrate how the refinement promoted by Hierarchical Visual Filtering deals with overlap of graphical items reducing the problems of visual clutter.

#### Problem formalization

Initially, we formalize the concepts of graphical item, screen-space, and screen-space coordinates. We consider that graphical
items refer to any distinct visual mark used for data visualization, screen-space refers to the available display space that a given visualization can use for presentation, and screen-space coordinates, in turn, refer to each position that can be addressed in a given screen-space.

The screen-space is naturally discrete in the number of coordinates, no matter whether the domain of the data being presented is discrete or continuous. Consequently, the number of coordinates in screen-space limits the number of graphical items that can be presented, what varies depending on the data domain that will be presented.

Although the number of possible graphical items can be much bigger than the number of screen-coordinates, the number of possible graphical items cannot exceed the number of screen-coordinates. This problem is even worse for graphical items that are bigger than one pixel. This limitation determines that visualization techniques suffer from overlap of graphical items and, consequently, by visual cluttering. In this context, overlap of graphical items is understood as the mapping of different data items to the same screen coordinate; meanwhile, visual cluttering refers to the confused or disordered arrangement that arises from overlap of graphical items, what negatively affects perception.

Overlap of graphical items
For this topic, we initially formalize the concept of region of interest. For a given visualization technique, a region of interest is a sub-area of the screen-space in which a collection of data items is being presented. The biggest region of interest is the whole visualization itself, meanwhile, smaller regions of interest are defined via interactive filtering.

Based on these notions, we can develop a formal metric for the overlapping of graphical items. The overlap can be measured by the density of elements that map to one same visual coordinate. Therefore, given a workspace $\Delta$ composed by the collection $D$ of data items, and by the set $G$ of graphical items, the overlap factor is given by the total number of elements mapped to $\Delta$, that is, $|D|$, divided by the total number of graphical items presented in $\Delta$, that is, $|G|$. Then:

$$overlap\_factor = \frac{|data\_elements\_mapped\_to\_\Delta|}{|graphical\_items\_presented\_in\_\Delta|} = \frac{|D|}{|G|} \quad (1)$$

The same holds for regions of interest. That is, given a region of interest $\delta \subset \Delta$, the overlap factor is given by the total number of elements mapped to this region, that is, $|d|$ for $d \subset D$, divided by the total number of graphical items presented in this region, that is, $g$ for $g \subset G$.

As presented in Figure 7, the Hierarchical Visual Filtering determines that, given a region of interest $\delta \subset \Delta$ with overlap factor $= |d|/|g|$, it is possible to create a new workspace $\Delta'$ with overlap factor $= |d'|/|g'|$. Considering that a region of interest is always smaller or equal to the screen-space, and considering that the workspaces created via pipelining use the whole screen-space, then, the following inequality is always valid $|g| \leq |G|$. This is because a wider screen-space will be available for the new pipelined workspace and, thus, more graphical items can be presented. Consequently, overlap factor $= overlap\_factor$ is always true. In other words, the pipelining operation will always create new scenes with equal or reduced overlap factor.

4 Experiments over pragmatic and epistemic actions
In this section we demonstrate the possibilities of the Hierarchical Visual Filtering; to do so, we perform experiments over a dataset of agrometeorological data - the Embrapa dataset. The dataset has 9 attributes: precipitation, maximum temperature, minimum temperature, normalized difference vegetation index (NDVI), water requirement satisfaction index (WRSI), average temperature, potential evapotranspiration (ETP), real evapotranspiration (ETR) and measured evapotranspiration (ETM) collected partly with remote sensors (satellite) and partly with in locus samples from sugar cane plantation regions in Brazil. The data was collected during 82 months from 2001 to 2007, so that each record corresponds to the data collected in a given month for one region. Since there are 5 regions: Araraquara (1), Araras (2), Jaboticabal (3), Jaú (4), and Ribeirão Preto (5), there is a total of 410 records. In order to test our hypothesis, we proceeded with the following tasks:

- characterize the 5 regions in relation to all the attributes of the dataset;
- choose two regions of interest in order to draw further visual clues comparatively.

The problem here is that the data is organized according to several regions and, although we want to inspect these regions separately, we do not want to use multiple visualization sessions. Multiple sessions would demand managing several windows, or even paper annotations. Also, the data is multivariate and it would be interesting to observe it according to multiple kinds of visualization techniques put together in a single environment. Lastly, it is desired to select subsets of the data and to summarize them by means of statistical calculations drawn over each of the visual workspaces, making the comparative analysis faster. In Figure 8, we show a visualization session for these data and for the proposed tasks; the image was edited for better presentation.

The visualization of all the data items over Parallel Coordinates – Figure 8(a) – presents a hard to interpret cluttering of lines. For this, the first step of the analytical process was to define 5 filtrations each corresponding to a different region. With the filtrations we could create 5 workspaces – Figure 8(b) – presented simultaneously and prone to detailed inspection by means of Instant Focus. Next, we want to characterize the workspaces’ data with a simple statistical operation - to do so, we calculate the average of each dimension at each workspace and proceed by drawing a polyline of average values emphasized in green on top of the Parallel Coordinates – also presented in Figure 8(b). This first round shows that the 5

![Figure 7: Treatment of graphical items overlap via pipeline refinement. (a) A workspace $\Delta$ and a region of interest $\delta \subset \Delta$ (black ellipse) defined with parameters $17.0 \leq ATTRIBUTE6 \leq 18.5$. (b) The workspace $\Delta'$ achieved via pipelining of the region of interest presented in (a). Also in (b), it is clear the increase in the number of graphical items, specially for ATTRIBUTE6. As consequence, in most dimensions not only the overlap factor was reduced but also the visual clutter in general.](image-url)
regions have similar values for each of the attributes with slight variations that confer them local peaks according to each average polyline.

In a next step, we want to visualize the same data in a multidimensional projection achieved using the well-known Fastmap algorithm; as so, in Figure 8(c), another visualization is presented for each of the workspaces. These new visualizations work by mapping the 9 attributes of the datasets into 3 dimensions, allowing one to inspect the data in respect to their dispersion in space, to the presence of clusters, and of outliers. The figure shows that region 1 (upper-most workspace) is the one whose data points better characterize a single cluster, contrasting with the others that define widely spread spatial regions. From this observation we assume that this region should be better analyzed, since evidence indicates that its data is homogeneous, potentially following a well-defined pattern. On the other hand, region 2 presents the plotting in which the points are more widely spread if compared to all the other workspaces, indicating a high-level of irregular behavior, what should be further investigated as well.

Next, we visually filter region 1 according to the 50-th percentile of attribute precipitation. This way we get a level deeper in the hierarchy and two new workspaces that we visualize according to techniques Parallel Coordinates and Scatter Plots. The first visualization accounts for the records of region 1 that present higher levels of rain – Figure 8(d), while the second accounts for the records with lower levels of rain – Figure 8(e). By inspecting the Parallel Coordinates average summary of both workspaces, there is a clear distinction in all of the attributes when the level of rain significantly varies. More specifically, the Scatter Plots visualization shows that the NDVI x Precipitation plot indicates that only part of the vegetation suffers with rain shortage, while a great part of it remains with high levels of green matter, possibly due to artificial irrigation. The same findings were verified for region 2, although not presented in the figure due to space limitations.

The demonstration of our system is not favored by the flat paper presentation; for this reason we put the system fully operational at http://gbdi.icmc.usp.br/~junio/VisTree/VisTree.htm, where it can be experimented with several other datasets.

User tests
In order to test the Hierarchical Visual Filtering in terms of usability, the same analytical tasks as described previously in this section were proposed to a group of 22 Computational Physics undergraduate students, familiar with computational tools. The students were instructed to use the VisTree system and to perform the same tasks, but in two different ways: in a first round, using Hierarchical Visual Filtering; in a second round, using one single workspace with simple visual filtering in order to create all the visualizations.
Under supervision, the students were monitored so that wall-clock time could be measured, and so that it would be possible to verify whether the tasks were really executed – without shortcuts. The experiment was conducted with the students, one at a time, in the same machinery, according to the following protocol:

- first, the students were trained with another dataset in order to get familiar with the system;
- the students received the expected answer of the analytical experiment: identify the two extreme regions, and to create further visualizations from each based on the precipitation attribute; this way, it would be possible to analyze only the Hierarchical Visual Filtering, and not the entire system;
- in the second round, the students received a different version of the dataset in which the labels of the regions differed from the dataset used in the first round; this way they would still have the challenge of identifying the two regions of interest;
- for the second round, the students could use paper to make annotations;
- for each task completed, the visualization was verified and the elapsed time was annotated.

In this experiment, 21 of the students were able to suitably complete the exercise. From these, all of them could complete the task faster by using the Hierarchical Visual Filtering, the gain was of 42% average – 4 minutes 52 seconds average for the first round, and 8 minutes 24 seconds average for the second round. Only 5 students used the paper for annotation, the remaining subjects preferred to use multiple windows and to alternate in between them. All of the subjects declared that the tasks were easier to be accomplished if Hierarchical Visual Filtering could be used.

For these results, we argue that the time gains would be higher for more complex tasks, or even for different tasks, instead of the same tasks performed with different operations; the same applies for the need of paper annotations. In despite of that, the results confirmed the adequacy and potential of the Hierarchical Visual Filtering analytically and empirically.

5 Conclusions
We have introduced Hierarchical Visual Filtering, an innovative interaction principle that overcomes cluttering issues and analytical complexity. Its principle is to allow the user to pipeline subsets of information to multiple workspaces that are automatically managed over a tree-like presentation. The essence of our contribution is based on actions that are pragmatic – filter and pipeline, and epistemic – record, annotate, and recall persistent visualizations.

We have performed experiments that demonstrate that our system- atization potentially speeds up and simplifies analytical tasks.

For now, we have experienced Hierarchical Visual Filtering over one single table (database relation); as further goals, we envision the design of ViSTree as a visual join tool, in which multiple related tables could be inspected. This design shall bring new challenges as join information may demand massive memory and processing resources; graphically, it will be necessary to adapt the Hierarchical Visual Filtering to a graph-like presentation, since a tree will not be enough; in terms of interaction, such configurations may be tricky as tables can be interrelated according to different cardinalities.

6 Acknowledgments
This work was partly supported by CNPq (Brazilian National Research Foundation), CAPES (Brazilian Committee for Graduate Studies), Microsoft Research and FAPESP (São Paulo State Research Foundation).

7 References