UMA HEURÍSTICA PARA O PROBLEMA DE CLASSIFICAÇÃO DE CONFERÊNCIAS EXPLORANDO RELACIONAMENTOS MÚLTIPLOS E INDIRETOS

GUILHERME HENRIQUE TRIELLI FERREIRA
Orientador: Wagner Meira Júnior

UMA HEURÍSTICA PARA O PROBLEMA DE CLASSIFICAÇÃO DE CONFERÊNCIAS EXPLORANDO RELACIONAMENTOS MÚLTIPLOS E INDIRETOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte, MG
Novembro de 2008

Ferreira, Guilherme Henrique Trielli

F383h Uma heurística para o problema de classificação de conferências explorando relacionamentos múltiplos e indiretos. / Guilherme Henrique Trielli Ferreira. — Belo Horizonte, 2008
xxi, 68 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Prof. Wagner Meira Júnior.

1. Recuperação da Informação - Tese. 2. Mineração de dados - Tese. 3. Multigrafos - Tese. 4. Grafos - Tese. 5. Redes sociais - Tese. 6. Heurísticas - Teses. I. Orientador. II. Título.

CDU 519.6*73(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Uma Heurística Baseada em Relacionamentos Múltiplos e Indiretos para o Problema de Classificação de Conferências

GUILHERME HENRIQUE TRIELLI FERREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. WAGNER MEIRA JÚNIOR - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. CRISTINA DUARTE MURTA
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET - MG

DRA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

PROF. MARCOS ANDRÉ GONÇALVES
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 12 de dezembro de 2008.

À Maria Alice, minha mãe, e à Renata Araujo, minha futura esposa, as duas mulheres de minha vida.

Agradecimentos

Gostaria de prestar aqui meus agradecimentos a todas as pessoas e instituições que, direta ou indiretamente, ajudaram-me a realizar este trabalho. Esta dissertação não é somente fruto de mais de dois anos de árduo trabalho, mas também de uma série de fatores presentes em minha vida.

Primeiramente, gostaria de agradecer aos meus pais, Willer Ferreira e Maria Alice Assumpção Trielli. A eles devo minha vida, minha educação e a oportunidade de buscar meus objetivos como venho fazendo. Querida mãe, sem seu monstruoso esforço para manter nossa casa e sem seu intenso amor, jamais estaria aqui. Esta é, sem dúvida, uma vitória para ambos nós. Ao restante de minha família, também agradeço e ressalto a importância de todos.

A próxima pessoa a agradecer chama-se Renata Braga Araújo, minha querida companheira. Desde o início de minha graduação até o final desse trabalho, ela foi crucial em minha pessoal e profissional. Seu amor e sua dedicação estarão impressos para sempre em minha alma. Agradeço também a sua família, que em um futuro breve será a nossa, por todo suporte e carinho dado a nós dois.

Em seguida, gostaria de agradecer aos meus amigos. Amigos de infância, como Igor dos Santos Fischer e Luis Felipe Cherem. Amigos que conheci na universidade, como Euder Flávio Rodrigues, Alison Carmo Arantes, Heitor Motta de Moraes, Luis Cunha Martins, Wagner Moro Aioffi, Luiz Guilherme dos Santos, entre outros. Essas amizades foram essenciais para manter minha sanidade durante períodos difíceis.

Apresento meus agradecimentos aos companheiros que tive no laboratório e-Speed e na empresa Synergia. Por muitas vezes essas pessoas me ajudaram em minhas atividades diárias, o que me permitiu manter o ritmo necessário para concluir este trabalho.

Gostaria de homenagear aqui também a uma lista enorme de professores, desde os do Colégio Magnum, que me ensinaram os fundamentos básicos das ciências, aos do Departamento de Ciência da Computação da UFMG, que abriram um novo mundo de conhecimento para a minha vida. Em especial, destaco os professores Wagner Meira Júnior e Renato Antônio Celso Ferreira, que foram meus orientadores durante os períodos de graduação e mestrado, e o professor Geraldo Robson Mateus, que por duas vezes

me deu a oportunidade de trabalhar na empresa Synergia.

Agradeço ao DCC e a UFMG por toda a maravilhosa infra-estrutura que tive durante tantos anos de intenso trabalho e aprendizado. Ressalto minha graditão e profundo respeito a instituição CNPq, que me concedeu uma inestimável bolsa de estudos, não somente durante o mestrado como também durante dois anos de minha graduação.

Existem ainda outros que não citei aqui, mas que estão em meu coração. A todos, meu muito obrigado.

Por fim, agradeço a Deus tudo que vivi e viverei.

"Se as coisas são inatingíveis... ora!
Não é motivo para não querê-las...
Que tristes os caminhos, se não fora
A presença distante das estrelas!"
(Mário Quintana)

Resumo

A habilidade de extrair conhecimento usável de grandes massas de dados tornou-se um dos mais importantes desafios para diversas comunidades, sejam elas científicas, industriais ou governamentais. A extração de tal conhecimento requer que os dados estejam representados de uma forma que não somente capture a informação relacional, mas também proporcione uma mineração efetiva e eficiente desses dados e uma capacidade de compreensão do conhecimento resultante. Na maioria dos casos, entretanto, os dados são modelados em grafos que não permitem a representação de relacionamentos múltiplos. Essa restrição, por sua vez, pode causar uma falha na capacidade de abstração entre os dados e o modelo, fazendo com que informações essenciais da aplicação real sejam completamente negligenciadas.

Este trabalho, portanto, versa sobre mineração de dados baseadas em relações, em que, em contraste a técnicas tradicionais, propomos uma heurística inovadora de mineração de multigrafos capaz de tratar relacionamentos múltiplos e indiretos nos dados, utilizando esses relacionamentos para identificar grupos correlacionados. Assim, construímos uma base teórica sobre a qual diversas aplicações reais podem ser modeladas, de forma a preservar relações importantes existentes nos dados, que eram ignoradas por outros modelos.

Aplicamos, então, a nossa técnica em um cenário real de redes de co-autoria, buscando agrupar e classificar fóruns científicos com base em afinidades de autoria. Para tal, modelamos os dados dessas redes como uma floresta de multigrafos e, então, os utilizamos para encontrar conjuntos de conferências que estejam interligadas. Caso tais conjuntos possam ser identificados em, pelo menos, um dado número de partes distintas da floresta, os mesmos serão vistos como pertencentes a uma mesma área de conhecimento. Resultados experimentais demonstram que a técnica é efetiva em extrair áreas distintas, mesmo diante de dados esparsos, a despeito do problema ser NP-Completo e o custo computacional da heurística ainda apresentar uma alta variabilidade.

Palavras-chave: Mineração de dados, Multigrafos, Agrupamento de dados, Redes de co-autoria

Abstract

Extracting usable knowledge from large amounts of data has become one of the main challenges to a variety of fields, such as scientific, industrial or governmental areas. This task requires the data to be represented in a way that not only is the relational information captured, but that it also allows an effective and efficient mining of these data and the understanding of the resulting knowledge. In most of the cases, however, the data are modeled as graphs that aren't able to represent multiple relations. This restriction may cause a flaw in the process of matching the real data with the model constructed, and, as a consequence, essential information of the real application is lost.

Therefore, this work discuss data mining based on relations where, in contrast with traditional techniques, we propose an innovative heuristic of multigraph mining capable of dealing with multiple and indirect relations in the data, using these relations to identify correlated groups. We constructed a theoretical base on which many real-life applications can be modeled, so that they preserve important relations that exist in the data and are ignored by other models.

We applied our new technique in a real scenario of co-authorship networks, in which we intend to group and classify scientific conferences based on authorship affinities. In order to do that, we modeled the data of these networks as multigraph sets, and then we use them to find groups of conferences that are correlated. If these groups can be found in, at least, a certain number of different parts of the multigraph, they will be considered as belonging to the same area. In spite of the fact that the problem we dealt with is NP-Complete and that there is a quite variety in the computational cost of the heuristic, experimental results show that our technique is effective in identifying different areas, even when the data is sparse.

Keywords: Data mining, Multigraphs, Clustering, Co-authorship networks

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O crescente acúmulo de informações em bancos de dados é um fato comum em diversas áreas do conhecimento, realçando a importância de sermos capazes de extrair conhecimento útil desses dados. Ao mesmo tempo, essa é uma tarefa não trivial, intrinsecamente dependente de uma alta capacidade computacional. A mineração de dados é justamente um conjunto de técnicas e ferramentas para a extração desse conhecimento, de forma que possamos utilizá-lo explicitamente para desenvolver ou aprimorar ações.

A habilidade de minerar ou extrair conhecimento usável tornou-se um dos mais importantes desafios para diversas comunidades, sejam elas científicas, industriais ou governamentais. Existem vários casos de sucesso quando os dados a serem explorados representam um conjunto de entidades independentes e seus atributos, como, por exemplo, transações comerciais. Entretanto, em muitos domínios (redes sociais, por exemplo), existe conhecimento relevante a ser minerado de relacionamentos entre entidades. A extração de tal conhecimento requer que os dados estejam representados de uma forma que não somente capture a informação relacional, mas também provenha uma mineração efetiva e eficiente desses dados e uma capacidade de compreensão do conhecimento resultante. A representação através de grafos, que consistem em coleções de ligações entre nodos, por sua vez, contém todos os aspectos do processamento de dados relacionais [4].

A mineração de grafos, portanto, visa extrair informações úteis de dados representados por um conjunto de nodos, onde seus pares podem ser conectados por arestas. O resultado dessa técnica é justamente um conjunto de padrões (subgrafos, no caso), que sejam freqüentes nesses dados, de acordo com um limiar de frequência preestabelecido. As aplicações da mineração de grafos abrangem diversas áreas, pois, em uma vasta gama de disciplinas, dados podem ser intuitivamente associados a essa representação. Por exemplo, redes de computadores são compostas por roteadores/computadores (nodos) e *links* (arestas) entre eles. Redes sociais são compostas por indivíduos e suas

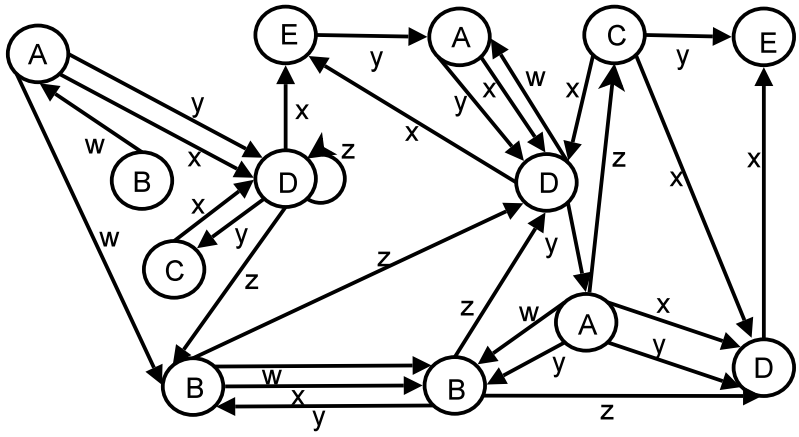

Figura 1.1. Exemplo de um multigrafo direcionado.

interconexões, que podem ser relações comerciais, de afinidade, de confiança etc. Redes de interação de proteínas ligam aquelas que podem trabalhar juntas para executar alguma função biológica.

Nos últimos anos, a mineração de dados aplicada sobre estruturas complexas, tais como grafos, árvores, moléculas, documentos *XML* e bancos de dados relacionais, atraiu uma grande quantidade de pesquisas. Padrões recorrentes em conjuntos de dados estruturados podem trazer à vista informações novas e vitais, escondidas exatamente nas relações que estão sendo modeladas. Quando os dados são considerados como um conjunto de entidades independentes e seus atributos, entretanto, essas informações podem ser completamente negligenciadas. Portanto, esses padrões recorrentes são um ponto de partida fundamental para pesquisas de mineração de dados com alto nível de qualidade, como a obtenção de agrupamentos e a geração de regras de classificação.

Na maioria dos casos, os dados são modelados em grafos que não permitem múltiplas arestas entre seus nodos. Entretanto, essa restrição pode causar uma falha na capacidade de abstração entre os dados e o modelo, pois podem existir múltiplos relacionamentos entre dois dos indivíduos sendo modelados. Por exemplo, em uma rede social, dois atores distintos podem ter mais de uma forma de interação, sendo ambas relevantes para o estudo da rede. Uma alternativa, então, é a modelagem dos dados em multigrafos (grafos que permitem mais de uma aresta entre dois nodos), direcionados ou não. Na Figura 1.1 vemos um exemplo de multigrafo. Nesse caso, temos um multigrafo direcionado, com rótulos nos nodos e nas arestas

Embora a mineração de dados estruturados possua uma ampla gama de problemas em que ela pode ser aplicada, existe um fator limitante fundamental: sua complexidade de tempo e espaço. A idéia de encontrar padrões freqüentes em grafos envolve a capacidade de reconhecer que um dado subgrafo está ou não contido em outro grafo,

o que é chamado de problema de isomorfismo de subgrafo. Ao contrário do problema de isomorfismo entre dois grafos, em que ainda não sabemos se ele está contido em P ou NP, o problema de isomorfismo de subgrafo é caracterizado como NP-completo [8]. Além do isomorfismo, outros fatores críticos característicos do problema de mineração de grafos são a intratabilidade do espaço de busca (exponencial em função do número de vértices) e a dificuldade de tratar todos os padrões resultantes (basta notar que para cada subgrafo freqüente, todas suas subestruturas são freqüentes). Como a maioria das aplicações envolvem grafos com um grande número de nodos e arestas, a mineração de multigrafos se caracteriza como um problema NP-completo, demandando o desenvolvimento de soluções através de heurísticas.

Dada a complexidade envolvida na mineração de grafos e o fato de essa ser uma área relativamente nova, as soluções existentes muitas vezes são insatisfatórias para aplicações reais. Dessa forma, o desenvolvimento de algoritmos para encontrar padrões complexos como grafos, de forma eficiente, constitui um esforço relevante.

O estudo de redes de citação na literatura científica [1] e o estudo de redes de co-autoria [16] destacam-se como exemplos relevantes da aplicação de mineração de grafos. As redes de co-autoria são formadas por um conjunto de autores (nodos) e um conjunto de relações de co-autoria de artigos publicados por esses autores (arestas). Embora tais redes sejam, geralmente, modeladas através de grafos simples, elas podem ser representadas com mais detalhe por multigrafos, onde existirá uma aresta ligando um autor a outro para cada conferência ou artigo no qual os dois tenham colaborado.

Em redes de co-autoria, existem vários problemas para os quais a mineração de multigrafos pode ser relevante. Citamos, entre eles, a classificação de conferências em áreas e subáreas; a descoberta de tendências em termos de publicações e parcerias entre autores; a evolução das conferências ao longo dos anos; e a previsão de relacionamentos, que pode, por exemplo, sugerir futuras parcerias entre autores. Neste trabalho, o foco é a classificação de conferências em áreas, mas a heurística desenvolvida também pode ser estendida para trabalhar com os outros problemas citados.

1.1 Objetivo

O objetivo desta dissertação é desenvolver uma heurística para o problema de agrupamento e classificação de conferências, através de uma abordagem inovadora que explora a modelagem como mineração de multigrafos. Dessa forma, criamos uma solução capaz de utilizar informações de relacionamentos múltiplos e indiretos presentes nos dados, que antes eram descartadas. Essa característica é uma vantagem, em comparação com estratégias tradicionais, uma vez que permite uma classificação mais abrangente.

1.2 Motivação

A principal motivação deste trabalho se baseia na necessidade de construção de técnicas de mineração de grafos que sejam capazes de modelar aplicações reais, cujos dados apresentam relacionamentos múltiplos e indiretos entre os indivíduos sendo modelados. Assim, padrões recorrentes em conjunto de dados estruturados podem trazer informações novas e essenciais, que antes eram escondidas por uma modelagem inapropriada.

Mesmo com tamanho potencial, a mineração de multigrafos é pouco explorada na literatura. A necessidade de novas técnicas para essa tarefa, especialmente dado a complexidade algoritmica do problema, constituiu um grande fonte de inspiração para o trabalho desenvolvido aqui.

A criação de uma heurística capaz de explorar esses relacionamentos indiretos (gerando agrupamentos, por exemplo), através da modelagem em multigrafos (como na aplicação de co-autoria e fóruns), se torna, assim, um tópico de pesquisa interessante. Usando multigrafos, o trabalho aqui apresentado produz resultados inovadores, pois sua abrangência e eficiência estão diretamente relacionadas à sua capacidade de explorar informações descartadas por outras técnicas.

1.3 Contribuições

As principais contribuições deste trabalho são:

- Modelagem de uma técnica inovadora de mineração de multigrafos capaz de tratar relacionamentos múltiplos e indiretos nos dados, utilizando esses relacionamentos para identificar grupos correlacionados. Essa contribuição, em especial, é fortalecida pelo fato do problema ser ainda pouquissímo explorado na literatura, não existindo técnicas semelhantes a apresentada aqui.

- Construção e definição de conceitos e fundamentos teóricos que permitem a estruturação dos dados, de forma que informações importantes e essenciais, intrínsecas a aplicações reais, fiquem em evidência para serem mineradas.

- Construção de uma base teórica sobre a qual diversas aplicações reais podem ser modeladas, de forma a preservar relações importantes existentes nos dados, que eram ignoradas por outros modelos.

- Utilização da técnica proposta em um cenário real: redes de co-autoria, comprovando sua usabilidade e eficácia na modelagem de problemas desse tipo.

1.4 Organização da dissertação

Neste capítulo inicial, fizemos uma breve introdução sobre os principais temas envolvidos nesta dissertação, destacando a área de aplicação dos mesmos. No capítulo a seguir, apresentamos uma revisão bibliográfica sobre os trabalhos encontrados na literatura relacionados a essa pesquisa, assim como definimos nosso escopo com maiores detalhes. Já o Capítulo 3 discorre sobre os fundamentos básicos (previamente definidos ou criados neste trabalho) necessários para o entendimento da heurística desenvolvida. O Capítulo 4 descreve detalhadamente nossa heurística, discutindo os pontos relevantes para seu desempenho em termos de eficiência de tempo de execução e qualidade dos resultados. Em seguida, o Capítulo 5 apresenta uma série de análises feitas a partir de diferentes experimentos realizados com o intuito de entender o comportamento da heurística. Por fim, o Capítulo 6 abrange as conclusões obtidas sobre os resultados desta dissertação, assim como possibilidades de futuros trabalhos que podem aprimorar e ampliar as utilidades da estratégia criada. As referências bibliográficas estão dispostas exatamente após o último capítulo.

Capítulo 2

A Mineração de Multigrafos e a Classificação de Conferências

Este capítulo descreve o problema tratado nesta dissertação, através de dois passos principais: a revisão bibliográfica sobre as principais áreas envolvidas e o estabelecimento do escopo deste trabalho em relação às mesmas.

2.1 Revisão bibliográfica

Nesta seção, apresentaremos algumas características das principais referências, presentes na literatura científica, envolvendo nosso trabalho. Dividimos tais pesquisas em dois tópicos: as técnicas de mineração de grafos e os trabalhos com redes de co-autoria e classificação de áreas. A seguir, descreveremos cada um deles em separado.

2.1.1 Mineração de grafos

Diferentes técnicas e metodologias estão sendo aplicadas para minerar padrões interessantes em forma de subgrafos, a partir de um banco de dados modelado na forma de grafos. Entre elas, podemos incluir abordagens de busca baseada na teoria matemática de grafos (como o FSG [14] e gSpan [25]), enfoques em buscas gulosas (como o Subdue [13] ou GBI [17]), trabalhos utilizando programação lógica indutiva ou ILP (como o WARMR [6]) e abordagens sobre bases indutivas (como o MolFea [23]).

Técnicas baseadas na teoria matemática de grafos mineram um conjunto completo de subgrafos usando principalmente uma medida de suporte ou freqüência. O trabalho inicial nessa área foi o sistema AGM [12], que usa o modelo de níveis do *Apriori*. Já o FSG possui uma estratégia similar e aperfeiçoa seu algoritmo para obter melhores tempos de execução. Ainda na mesma área, o gSpan usa códigos retirados de buscas

em profundidade para gerar rótulos canônicos, sendo muito mais eficiente em termos de memória e poder computacional do que as abordagens prévias. Podemos citar também o CloseGraph [26] que, em vez de minerar todos os subgrafos, minera apenas os "subgrafos fechados". Um grafo G é considerado fechado em um conjunto de dados se não existe algum supergrafo de G com o mesmo suporte do próprio G. O algoritmo Gaston [21], por sua vez, minera eficientemente conjuntos de grafos, determinando, primeiramente, os caminhos freqüentes que são transformados em árvores, para, em seguida, transformá-los em grafos. Por fim, o FFSM [11] é um sistema de mineração de grafos que usa um *framework* algébrico para endereçar o subproblema de isomorfismo de grafos.

Além de abordagens que utilizam a teoria matemática de grafos, existem também aquelas baseadas em buscas gulosas, que usam heurísticas para avaliar a solução. Dois trabalhos pioneiros nesse campo são o Subdue e o GBI. O Subdue utiliza heurísticas de compressão baseada em MDL (*minimum description length*), enquanto o GBI usa uma heurística empírica baseada no tamanho do grafo. A definição empírica do tamanho do grafo depende do tamanho dos padrões extraídos e também do tamanho do grafo comprimido.

Outra metodologia de mineração de grafos envolve programação lógica indutiva, que possui a vantagem do poder de descrição da lógica de primeira ordem. O primeiro sistema baseado em grafos que combinou ILP e o método de busca do *Apriori* foi o WARMR (tal sistema foi utilizado sobre um banco de dados de componentes químicos). A grande vantagem desse tipo de enfoque é o seu alto poder de representação.

Apesar de alguns dos trabalhos mencionados aqui defenderem a idéia de que seus algoritmos podem ser expandidos para lidar com multigrafos, nenhum deles trabalha diretamente sobre esse tipo de entrada. Assim, suas estratégias, embora eficientes para grafos específicos, não possuem nenhuma especialização voltada para multigrafos. Isso deixa esse campo de pesquisa aberto para o desenvolvimento de tais técnicas, sendo o presente trabalho justamente uma alternativa para explorar diretamente os múltiplos relacionamentos entre dados.

2.1.2 Classificação de conferências

Redes de co-autoria são um exemplo importante das redes sociais e vêm sendo extensamente utilizadas para determinar a estrutura de colaborações científicas e a categoria (ou *status*) individual de conferências e/ou pesquisadores. Embora parecidas com as muito estudadas redes de citação na literatura científica [9], redes de co-autoria representam um elo social muito mais forte do que uma citação. Citações podem ocorrer sem que os autores se conheçam e acontecem sem um padrão temporal. Já as redes

de co-autoria representam uma relação acadêmica e temporal, o que a qualifica mais claramente para a análise de redes sociais.

Um exemplo de rede de co-autoria é o *Erdös Number Project*, na qual é calculado o menor número de ligações de co-autoria entre qualquer matemático e o matemático húngaro Erdös [2]. Newman, por sua vez, estudou e comparou o grafo de co-autoria de arXiv, Medline, SPIRES e NCSTRL [19, 20] e encontrou muitas diferenças entre redes de disciplinas experimentais e teóricas. A análise de co-autoria tem sido aplicada também a várias conferências da *ACM - Information Retrieval* (SIGIR) [24], *Management of Data* (SIGMOD) [18] e *Hypertext* [3] - e também às áreas de matemática e neurociência [7], sistemas de informação [5] e análise de redes sociais [22]. Além disso, redes de co-autoria internacionais foram estudadas no *Journal of American Society for Science & Technology* [10] e *Science Citation Index* [15].

2.2 O contexto do trabalho

Uma das possíveis aplicações da mineração de multigrafos às redes de co-autoria é a capacidade de classificar as conferências segundo áreas. Definir de qual área uma conferência faz parte é uma tarefa importante e pode ser usada a fim de, por exemplo, indicar a um autor em quais conferências ele deve tentar publicar seu mais recente trabalho.

Para respondermos a tal pergunta, precisamos definir qual a nossa concepção de área. Uma área pode ser vista como um conjunto de temas que envolvem problemas muito semelhantes - por exemplo, computação móvel, mineração de dados, bibliotecas digitais - sendo que as conferências normalmente são especializadas para alguma determinada área ou subárea. Entendemos também que os autores tendem a publicar artigos sobre temas envolvidos nas áreas em que ele já realizou trabalhos prévios. Dessa forma, podemos ver uma área como um conjunto de conferências em que diversos autores distintos concentram suas publicações.

A heurística apresentada neste trabalho utiliza a estratégia de busca em profundidade. Nesse ponto, ele é comparável com os trabalhos como o gSpan. Ao mesmo tempo, buscamos desenvolver uma heurística gulosa, o que o aproxima do Subdue. Entretanto, duas diferenças são fundamentais, quando comparamos nosso trabalho com os demais citados: trabalhamos especificamente com multigrafos e não nos preocupamos com a topologia desses padrões, dado o objetivo de classificar conferências.

A modelagem através de multigrafos para esse tópico traz algumas vantagens claras. Primeiramente, permite que as relações de co-autoria sejam tratadas individualmente de acordo com o veículo ou fórum, já que dois nodos podem possuir várias arestas entre

eles, uma para cada relacionamento partilhado. Dessa forma, é reduzida a possibilidade de ruídos nos resultados, devido a, por exemplo, autores que publicam em muitas áreas diferentes. A segunda vantagem que podemos citar é que, com múltiplas arestas, a análise dos subgrafos pode ser feita diretamente, ao contrário do que aconteceria se tentássemos representar uma combinação de rótulos em uma única aresta, em um grafo convencional. Outro aspecto interessante é relativo ao uso das técnicas desenvolvida neste trabalho. Conforme discutiremos, nossa estratégia perde a informação de topologia para quaisquer padrões com mais de duas arestas. Entretanto, neste problema, nos interessa analisar apenas os rótulos de arestas que estão presentes em vários conjuntos de nodos distintos, através de uma relação global e não local. Assim, temos que essa estratégia de solução é promissora em termos da qualidade da definição das áreas, pois encontra justamente conjuntos de conferências nos quais usualmente os autores interagem entre si.

Nesta dissertação, trabalharemos aplicando a mineração de multigrafos em redes de co-autoria. A intenção é que essa estratégia nos permita descobrir informações de forma inovadora, como, por exemplo, características das comunidades de uma rede de co-autoria, através da análise da frequência que um dado perfil de autor publica artigos em conjunto com autores de outros perfis.

Dessa forma, apresentaremos, no capítulo a seguir, um conjunto de fundamentos teóricos essenciais para a heurística desenvolvida.

Capítulo 3

Fundamentos Básicos

Conforme citamos previamente, para encontrarmos padrões freqüentes em forma de multigrafos, propomos uma heurística baseada na busca em profundidade. Este capítulo concentra-se, então, na definição dos conceitos básicos que irão ser usados para nortear a construção da nossa heurística. Dessa forma, serão formalizados os seguintes conceitos: grafos e multigrafos (ambos não direcionados); trincas e suas instâncias; cadeias de trincas e os conjuntos de mapeamentos respectivos; árvore de trincas; e, por fim, cadeias freqüentes.

3.1 Definição de multigrafos não-direcionados

Neste trabalho, exploramos o agrupamento de dados modelados como um multigrafo, um tipo especial de grafo não-direcionado que permite a ocorrência de mais de uma aresta entre dois vértices quaisquer. Definiremos agora, formalmente, os conceitos de grafo e multigrafo.

Dado um alfabeto de rótulos R, podemos definir um grafo não direcionado g como uma quádrupla $g = (V, A, l_v, l_a)$, onde:

- V define um conjunto de vértices.
- $A \subseteq V \times V$ define um conjunto de arestas.
- l_v: $V \rightarrow R' \subseteq R$ define uma função de rótulos para os vértices.
- l_a: $A \rightarrow R'' \subseteq R$ define uma função de rótulos para as arestas.

O conjunto V pode ser visto como o conjunto de identificadores de vértices, sendo $V = \{1, 2, 3, ..., |V|\}$. Enquanto V identifica os vértices, o conjunto de arestas A representa a estrutura do multigrafo. Isto é, um vértice $u \in V$ está conectado a outro vértice $v \in V$

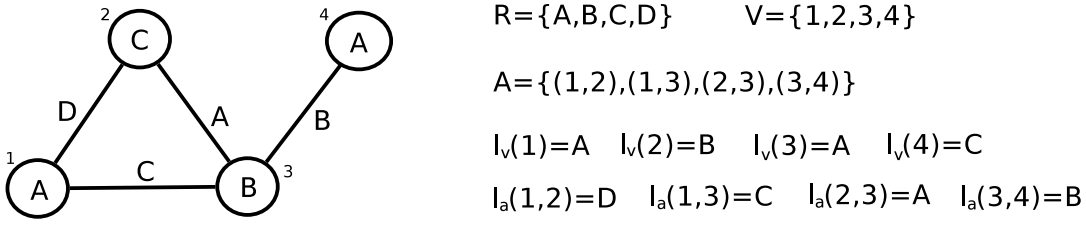

Figura 3.1. Exemplo de um grafo não direcionado.

por uma aresta $e = (u, v)$ se $(u, v) \in A$. Ainda temos que as arestas do grafo não são direcionadas, logo, para todo $(u, v) \in A$ temos $(v, u) \in A$, tal que $l_a(u, v) = l_a(v, u)$. Por fim, ressaltamos que os conjuntos R' e R'' não precisam possuir nenhuma relação, sendo totalmente independentes. Ambos foram definidos como subconjuntos de R apenas por conveniência, pois mais a frente iremos supor uma ordenação natural entre os todos os elementos desse último conjunto.

A Figura 3.1 apresenta um exemplo simples de um grafo $g = (V, A, l_v, l_a)$, que utiliza um conjunto de rótulos R. As arestas redundantes, provenientes do bi-direcionamento dos relacionamentos, foram omitidas para facilitar o entendimento (recurso utilizado em diversas figuras ao longo desta dissertação).

Um multigrafo, por sua vez, é um grafo que permite a ocorrência de mais de uma aresta entre dois vértices quaisquer. Logo, precisamos alterar a formalização anterior da seguinte forma: um multigrafo m_g pode ser definido como uma tripla $m_g = (V, A, l_v)$, onde:

- V define um conjunto de vértices.
- $A \subseteq V \times V \times R'' \subseteq R$ define um conjunto de arestas.
- l_v: $V \rightarrow R' \subseteq R$ define uma função de rótulos para os vértices.

A grande diferença entre as duas definições é relativa ao conjunto A, no qual, na segunda definição, adicionamos o rótulo à identidade de uma aresta. Em outras palavras, permitimos que para dois vértices $u, v \in V$, tenhamos até $|R|$ arestas válidas distintas, como, por exemplo, (u, v, r_1) e (u, v, r_2), bastando que $r_1 \neq r_2$, $r_1, r_2 \in R$. Por fim, o conceito de arestas não direcionadas implica que para todo $(u, v, r_1) \in A$ temos $(v, u, r_1) \in A$. Um exemplo de multigrafo não direcionado $m_g = (V, A, l_v)$ é dado na Figura 3.2, que utiliza um conjunto de rótulos R. As arestas redundantes, devido ao bi-direcionamento dos relacionamentos, foram novamente omitidas. É interessante notar que, pelas definições acima, todo grafo é também um multigrafo, mas a recíproca não é verdadeira.

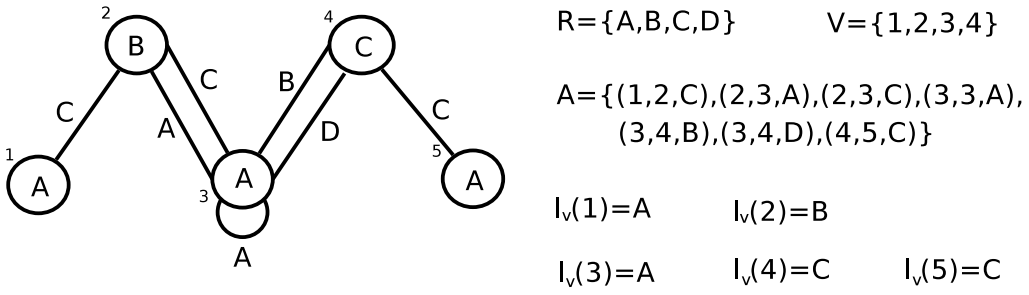

Figura 3.2. Exemplo de um multigrafo não direcionado.

3.2 Trincas e seu conjunto de instâncias

Em nosso algoritmo de agrupamento, os dados são modelados em um multigrafo, com rótulos nos nodos e nas arestas. Tais rótulos são "qualificadores" desses dados, ou seja, definem o tipo dos elementos sendo relacionados. Portanto, nesta seção, procuramos criar uma estrutura que condensasse os dados de uma aresta do multigrafo, justamente através dos rótulos dessa mesma aresta e dos nodos conectados por ela. Nossa estratégia é, então, agrupar as informações, de acordo com seus relacionamentos.

Para que fosse possível explorar tais características, definimos uma trinca $t = (r_1, r_2, r_3)$, de forma que t será uma trinca válida para o multigrafo m_g se, e somente se, existir uma aresta $(u, v, r) \in A$, tal que:

- $l_v(u) = r_1$
- $l_v(v) = r_2$
- $r = r_3$

Dessa maneira, a trinca será um elemento básico que contém as seguintes informações: os rótulos dos nodos e o rótulo da aresta que os relaciona. Lembramos novamente que não existe nenhuma relação entre rótulos de nodos e arestas.

Assim, como nodos diferentes podem receber um mesmo rótulo, note que uma trinca pode representar várias arestas diferentes, onde denotaremos cada uma delas de instância da trinca em questão. No multigrafo da Figura 3.2, por exemplo, temos a trinca (A, B, C) que representa as arestas $(1, 2, C)$ e $(2, 3, C)$. Dessa forma, para uma trinca t, temos como seu conjunto de instâncias $I(t) \subseteq A$. No exemplo dado, $I(A, B, C) = \{(1, 2, C), (2, 3, C)\}$. O número de elementos desse conjunto será o fator chave quando formos considerar a relevância da trinca, conforme explicaremos durante a discussão sobre a heurística deste trabalho.

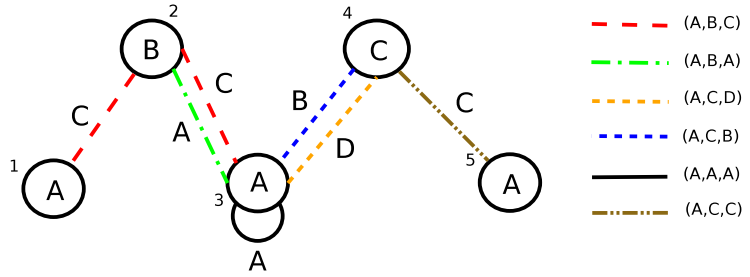

Figura 3.3. Trincas do multigrafo da Figura 3.2 destacadas.

Pela definição, uma aresta do grafo será associada a uma, e somente uma, trinca. Esse fato implica que, dadas duas trincas distintas t_i e t_j, $I(t_i) \cap I(t_j) = \emptyset$. Assim, é fácil perceber que caso existam n trincas distintas, temos que $U_{i=1}^{n} t_i = A$. Analisando novamente a Figura 3.2, temos como trincas: (A, A, A), (A, B, A), (A, B, C), (A, C, B), (A, C, C) e (A, C, D). A união do conjunto de instâncias das mesmas é igual ao conjunto de arestas do multigrafo. Na subseção a seguir, explicamos porque representamos essa trinca como (A, B, C) e não como (B, A, C). Na Figura 3.3, apresentamos tais trincas do multigrafo original, em destaque.

Acreditamos que uma trinca expressa um relacionamento de uma maneira genérica, porém exata. Genérica, pois, como descrito acima, ela pode representar distintas arestas do grafo. Ela também é exata, posto que define de maneira única cada relacionamento presente no multigrafo, definido pela "classe" dos dados envolvidos.

3.3 Comparação entre trincas

Dado um conjunto de trincas distintas, um ponto chave do nosso algoritmo é como relacioná-las, de forma a obtermos um padrão freqüente. Iremos, então, definir um conceito de ordenação total para as trincas, obtendo uma forma de comparar duas trincas distintas quaisquer.

Como uma trinca é formada por três elementos de R (o conjunto de rótulos formado pela união de R' e R'', conforme Seção 3.1), vamos supor uma ordenação natural entre os elementos de R, baseada na ordenação lexicográfica da cadeia de caracteres que os descreve. É importante ressaltar que essa opção foi escolhida apenas por sua simplicidade, podendo ser substituída por outra forma que também tenha a propriedade de ordenação total entre os elementos. A partir dessa ordenação, estabelecer que uma trinca $t = (r_1, r_2, r_3)$ será sempre representada por (r_1, r_2, r_3), se $r_1 \leq r_2$, ou (r_2, r_1, r_3), caso contrário.

Assim, duas trincas $t_1 = (r_1, r_2, r_3)$ e $t_2 = (r_4, r_5, r_6)$ podem estar relacionadas da

seguinte forma:

- $t_1 = t_2$, se $r_1 = r_4$, $r_2 = r_5$ e $r_3 = r_6$.
- $t_1 < t_2$, se $r_1 < r_4$, ou se $r_1 = r_4$ e $r_2 < r_5$ ou se $r_1 = r_4$, $r_2 = r_5$ e $r_3 < r_6$.
- $t_1 > t_2$, para os demais casos.

Para o multigrafo da Figura 3.3, temos então a seguinte relação:

$$(A, A, A) < (A, B, A) < (A, B, C) < (A, C, B) < (A, C, C) < (A, C, D)$$

.

A partir desse momento, faz sentido dizer que uma trinca é maior ou menor que outra. Essa afirmação não subentende nenhuma informação sobre a trinca, mas é fundamental para definirmos ferramentas que relacionam essas entidades, conforme veremos nas seções a seguir.

3.4 Cadeias de trincas

Cada trinca, conforme a definição apresentada, pode ser vista como um relacionamento único presente no multigrafo. Como nossa intenção é encontrar um conjunto de relacionamentos que é significativo (para podermos encontrar áreas de conhecimento, no trabalho de redes de co-autoria), resta definir como construir cadeias de trincas e avaliá-las.

Uma cadeia de trincas c de tamanho n ($n > 0$) é, simplesmente, um multiconjunto (um conjunto que permite elementos repetidos) de trincas, cuja cardinalidade será n, mas cujo número total de trincas distintas poderá ser entre 0 e n. Utilizaremos a notação $c(t_1, t_2, ..., t_n)$ ou $c = \{t_1, t_2, ..., t_n\}$ para representar uma cadeia formada por n trincas t_i. Dessa forma, a partir de um conjunto T de trincas, podemos definir infinitas cadeias distintas. Na Figura 3.4 apresentamos alguns exemplos de cadeias de trincas possíveis para o multigrafo da Figura 3.3, que possui como conjunto de trincas $T = \{(A, A, A), (A, B, A), (A, B, C), (A, C, B), (A, C, C), (A, C, D)\}$.

Dada uma cadeia $c = \{t_1, t_2, ..., t_n\}$, podemos, agora, criar o conceito de mapeamento dessa cadeia, que é análogo ao conceito de instância de uma trinca. Cada mapeamento m de c será também um conjunto de arestas que atende às duas restrições a seguir:

- m tem a forma $m = \{e_1, e_2, ..., e_n\}$, onde $\forall ti \in c$, $\exists e_i \in m$, $e_i \in I(t_i)$. Ou seja, para cada trinca da cadeia, existe uma, e somente uma, aresta correspondente no mapeamento.

c_1 = { (A,B,C) }

c_2 = { (A,B,A), (A,C,D) }

c_3 = { (A,B,A),(A,C,B) }

c_4 = { (A,A,A),(A,B,C),(A,B,C) }

c_5 = { (A,A,A),(AAA),(A,B,A),(A,B,C),(A,C,D) }

Figura 3.4. Exemplos de cadeias de trincas.

- para duas arestas e_i e e_j quaisquer de m, existe um caminho composto apenas por elementos de m, que liga os vértices de ambas as arestas no multigrafo original.

A primeira restrição mostra que um mapeamento é, de forma simplificada, um conjunto de arestas que são instâncias das trincas que formam a cadeia. A segunda indica que, se juntarmos as arestas de um mapeamento, teremos um subgrafo conectado do multigrafo de origem. Por isso, ao contrário da cadeia, que é um multiconjunto, o mapeamento da cadeia é um conjunto, pois a repetição de uma mesma aresta em um subgrafo não teria significado algum.

m_1= { (2,3,A),(1,2,C) } m_2= { (2,3,A),(2,3,C) }

Figura 3.5. Múltiplos mapeamentos de uma mesma cadeia.

Na Figura 3.5, apresentamos um exemplo de dois mapeamentos distintos da cadeia $c = \{(A, B, A), (A, B, C)\}$ formada a partir do multigrafo da Figura 3.3. É interessante perceber que os mapeamentos de uma mesma cadeia podem representar subgrafos de topologias distintas, porém que englobam o mesmo conjunto de relacionamentos. Um ponto a ser ressaltado é que, tanto em cadeias quanto em mapeamentos(por serem multiconjuntos e conjuntos, respectivamente) não há ordenação entre seus elementos.

Note que um mapeamento está associado a apenas uma cadeia, mas uma cadeia pode possuir vários mapeamentos distintos. Chamaremos de $M(c)$ o conjunto de todos os mapeamentos de uma cadeia de trincas c. Assim, dado um multigrafo, apenas as cadeias c tais que $M(c) \neq \emptyset$ serão consideradas válidas. Isso porque, uma cadeia que

M({(A,B,A),(A,B,C),(A,B,C)})={ {(2,3,A),(2,3,C),(2,1,C)} }

M({(A,B,C)})={ {(1,2,C)},{(2,3,C)} }

M({(A,C,B),(A,C,C)})= ∅

M({(A,A,A),(A,B,A),(A,C,C)})= ∅

Figura 3.6. Algumas cadeias de trincas e seus mapeamentos.

não pode ser associada a um subgrafo conectado não terá utilidade em nossos estudos sobre características de multigrafos específicos. Na Figura 3.6 apresentamos exemplos de cadeias e seus mapeamentos, para o multigrafo da Figura 3.3. Conforme as definições acima, as duas primeiras são válidas e as outras duas não.

Para finalizar essa seção sobre encadeamentos de trincas para a construção de cadeias, iremos definir uma operação simples, denominada de expansão de cadeia. Dadas uma cadeia $c_n = (t_1, t_2, ..., t_n)$ e uma trinca t_i, a expansão da cadeia c_n com a trinca t_i será uma operação que produz uma nova cadeia $c' = (t_1, t_2, ..., t_n, ti)$. O Algoritmo 3.1 é um pseudo-algoritmo para construirmos $M(c')$ a partir do conjunto de mapeamentos $M(c_n)$ e do de instâncias $I(t_i)$. A condição apresentada na linha seis garante que o novo mapeamento atenderá às restrições citadas nessa seção. Já as condições das linhas cinco e oito precisam ser utilizadas apenas quando permite-se que uma cadeia possua trincas repetidas. No caso de nossa aplicação, que envolve agrupamento e classificação de fóruns, essas condições não existirão, pois os agrupamentos não permitem repetições, conforme veremos nos capítulos posteriores.

3.5 Árvore de trincas

Anteriormente, definimos trincas e como encadeá-las. Agora, precisamos de uma ferramenta para estruturar as diferentes cadeias de trincas possíveis, de forma que o maior número possível de estruturas distintas do multigrafo sejam exploradas. Essa ferramenta será a árvore de trincas.

Dado um multigrafo $m_g = (V, A, l_v)$, definimos para o mesmo uma árvore de trincas como uma árvore padrão (um grafo simples, conectado e acíclico) onde todos os nodos terão como rótulo uma trinca válida para m_g, exceto a raiz, que não possui rótulo. A principal característica da árvore de trincas é que cada nodo está associado à cadeia formada pelos rótulos de todos os nodos no caminho entre tal nodo e a raiz da árvore. Além disso, só existe um caminho que forma essa cadeia. Na Figura 3.7, mostramos

Entrada: $c_n, M(c_n), t_i, I(t_i)$
Saída: $M(c')$

```
1  início
2      M(c') ← ∅;
3      para cada mapeamento m_k ∈ M(c_n) faça
4          para cada instância i_q ∈ I(t_i) faça
5              se i_q ∉ m_k então
6                  se m_k possui um dos dois índices de nodo de i_q então
7                      e ← m_k ∪ {i_q};
8                      se e ∉ M(c') então
9                          Insira e no conjunto M(c');
10     retorna M(c');
11 fim
```

Algoritmo 3.1: Definindo os mapeamentos de uma cadeia em uma expansão.

um exemplo de uma possível árvore de trincas a ser formada a partir do multigrafo da Figura 3.2. O nodo 3 representa a cadeia $c_3 = \{(A, C, D)\}$, o nodo 5 representa a cadeia $c_5 = \{(A, A, A), (A, B, C)\}$ e o nodo 8, a cadeia $c_8 = \{(A, B, C), (A, C, D), (A, C, D)\}$.

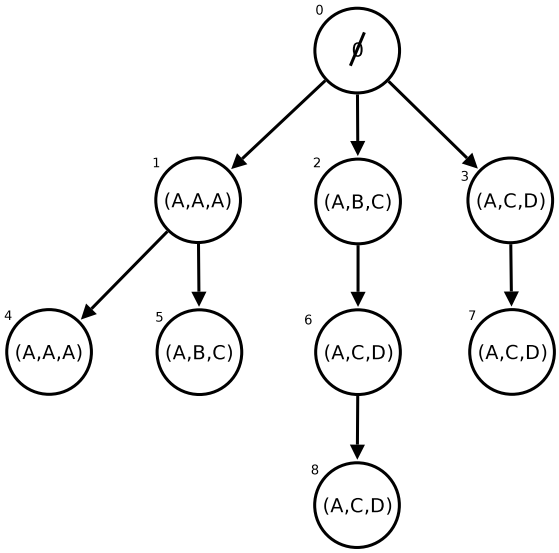

Figura 3.7. Exemplo de uma possível árvore de trincas.

Para garantirmos que dois nodos distintos não sejam associados à mesma cadeia, todos os nodos devem atender a duas condições:

1. dois irmãos devem, necessariamente, possuir rótulos distintos.

2. os rótulos dos filhos não podem ser menores que o rótulo do pai.

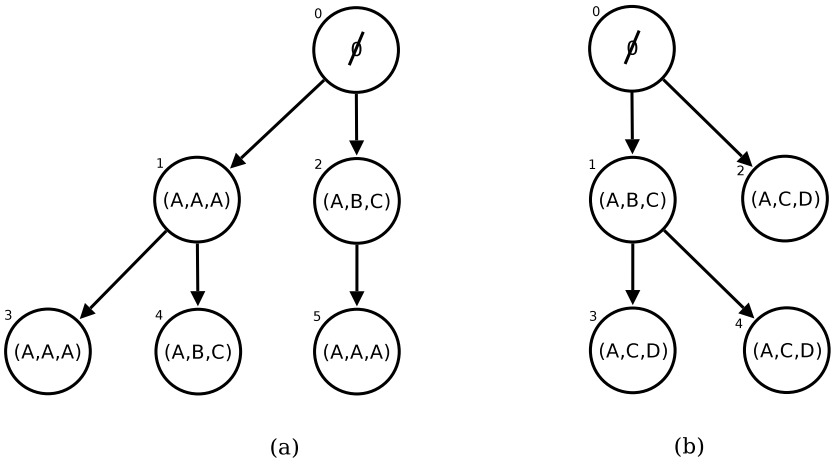

Figura 3.8. Duas árvores de trincas inválidas.

Na Figura 3.8, apresentamos duas árvores inválidas para o multigrafo da Figura 3.2, pois ambas não atendem as restrições citadas acima. Note que, para a árvore (a), o nodo 5 possui como rótulo um trinca que é menor que o rótulo de seu pai (o nodo 2) e, dessa forma, representa a mesma cadeia que o nodo 4. Já para a árvore (b), os nodos irmãos 3 e 4 possuem o mesmo rótulo, fazendo com que ambos identifiquem a mesma cadeia.

Dessa forma, como cada nodo n_i da árvore está associado exclusivamente a uma cadeia c_i, podemos também associar a n_i o conjunto de mapeamentos $M(c_i)$. Apesar de essa formalização não possuir utilidade no conceito de árvores de trincas, ela será útil quando apresentarmos as formas de operação da heurística sobre sua estrutura.

Dado um multigrafo m_g, cujo conjunto de trincas seja T, a árvore de trincas que representa todas as cadeias possíveis (denominada árvore completa) é aquela que cada nodo i (de trinca t_i) tenha um filho para cada $t_j \in T$ que obedeça a restrição $t_i \leq t_j$. Note que essa árvore terá profundidade infinita, pois existem infinitas cadeias. Portanto, trabalharemos com o conceito de cadeias válidas para definir uma árvore de trincas válidas.

Uma árvore de trincas válidas será aquela cujos nodos, com exceção da raiz, possuam associados a si uma cadeia válida. Conforme definição prévia, uma cadeia é válida se seu conjunto de mapeamentos não for vazio, sendo que um mapeamento não pode ter arestas repetidas. Assim, é fácil perceber que uma árvore que possua todas as trincas válidas possíveis será um tipo especial de árvore completa, onde adiciona-se a restrição que um filho, de trinca t_j como rótulo, será inserido em um nodo apenas se representar uma cadeia válida. Ao contrário da árvore completa, essa árvore tem profundidade finita, pois o número de arestas de um multigrafo também é finito. É importante

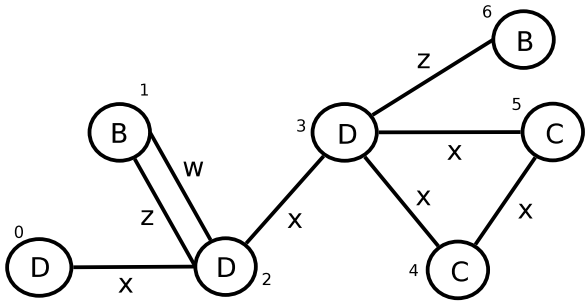

Figura 3.9. Outro exemplo de multigrafo.

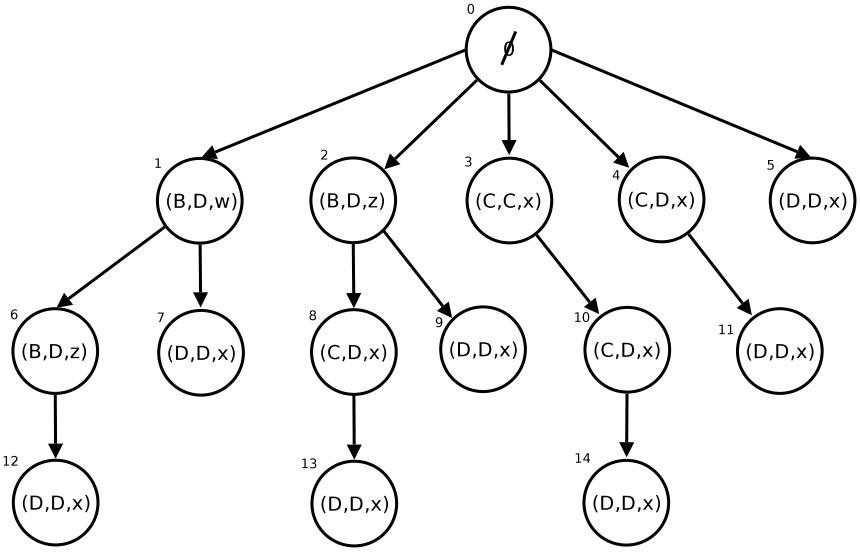

Figura 3.10. Uma árvore de trincas válidas completa.

ressaltar que, segundo essa definição conceitual, para qualquer multigrafo m_g podemos derivar uma árvore de trincas válidas completa.

Na Figura 3.10, uma árvore de trincas válidas onde todas as possibilidades de cadeias foram exploradas, derivada a partir do multigrafo da Figura 3.9. Nesse exemplo, não permitimos a repetição de trincas nas cadeias. Note que, conforme a definição, nenhuma folha pode ser expandida de forma a gerar uma cadeia válida.

3.6 Cadeias freqüentes

Nesta seção, definimos o que é uma cadeia de trincas freqüente. Esse conceito é fundamental para a criação da heurística, pois estamos interessados apenas em cadeias que são freqüentes.

O primeiro passo é estabelecer o que é o suporte de uma cadeia c, da seguinte forma: o suporte $S(c)$ é dado por $|M(c)|$, ou seja, é igual ao número de mapeamentos existentes para essa cadeia. Assim, definimos que c será freqüente se, e somente se:

$$S(c) \geq \lambda,$$

onde λ é um número inteiro pré-estabelecido, também chamado de suporte mínimo.

Como exemplo, analisaremos algumas cadeias representadas pelos nodos da árvore da Figura 3.10. Se considerarmos $\lambda = 2$:

- a cadeia dada pelo nodo 9, $c_9 = (B, D, z), (D, D, x)$, é freqüente, pois $M(c_9) = \{\{(1, 2, z), (2, 3, x)\}, \{(6, 3, z), (2, 3, x)\}\}$, logo $|M(c_9)| = 2 \geq \lambda$. De forma análoga, as cadeias dos nodos 2, 4, 5, 8, 10, 11 e 13 e também são freqüentes.

- a cadeia dada pelo nodo 6, $c_6 = (B, D, w), (B, D, z)$, não é freqüente, pois $M(c_6) = \{\{(1, 2, w), (1, 2, z)\}\}$, logo $|M(c_6)| = 1 < \lambda$. Pela mesma razão, não são freqüentes também as cadeias representadas pelos nodos 1, 3, 7, 12 e 14.

Esse conceito de freqüência é importante, pois pode ser utilizado durante a construção de árvores de cadeias válidas. Ou seja, ao invés de mantermos na árvore cadeias válidas (com pelo menos um mapeamento), podemos manter cadeias freqüentes (com pelo menos λ mapeamentos). É com base nessa idéia que definimos nossa heurística.

A seguir, então, utilizamos todos os fundamentos, definidos ao longo desse capítulo, para descrevermos a heurística desenvolvida neste trabalho. Nossa técnica é construída para ser capaz de explorar a árvore de cadeias freqüentes completa gerada a partir de um multigrafo qualquer, tentando encontrar o maior número possível de padrões freqüentes.

Capítulo 4

A Heurística VL

Definimos, agora, a heurística desenvolvido nesta dissertação para promover o agrupamento dos dados, utilizando a informação de relacionamentos múltiplos e indiretos. Essa heurística será referenciada, desse momento em diante, como heurística da Vizinhança Limitada, ou simplesmente, VL.

4.1 Algoritmo básico

Através dos conceitos envolvendo trincas e cadeias, definidos anteriormente, a heurística VL busca promover uma mineração de subgrafos em um multigrafo, para encontrar agrupamentos. Assim, podemos definir nosso algoritmo da seguinte forma: dado um multigrafo, encontramos diversos conjuntos de subgrafos distintos (representam a mesma cadeia de trincas, porém com topologias diferentes) que ocorram com uma freqüência de ocorrência maior que um dado λ. Isso significa que o algoritmo busca retornar todas as cadeias cujo conjunto de mapeamentos possua, no mínimo, λ elementos.

Esse algoritmo foi construído baseado em uma estratégia gulosa, ou seja, possui etapas seqüenciais que buscam um resultado ótimo local, porém o resultado ótimo global não é garantido. Uma maneira de explicar nossa heurística é descrevê-la como uma busca em profundidade na árvore completa de cadeias válidas. Assim, podemos dividí-la em duas etapas: a fase inicial, onde encontramos quais cadeias, com apenas uma trinca, são freqüentes; e a fase de construção das cadeias com duas ou mais trincas, a partir da expansão das mesmas com as trincas das cadeias de tamanho um.

O Algoritmo 4.1 apresenta a seqüência de passos do nosso algoritmo. As principais entradas são a freqüência mínima λ e a base de dados já modelada em um multigrafo. A linha 2 representa uma leitura dos dados, de forma a identificar quais são as trincas existentes e seus mapeamentos. Em seguida, a linha 3 cria a raiz da árvore de trincas

```
   Entrada: λ, m_g
   Saída: Cadeias de trincas freqüentes
1  início
2      T ← EncontraTrincas(m_g);
3      Inicialize a árvore de trincas com a raiz r;
4      para cada trinca t_i ∈ T faça
5          se |M({t_i})| ≥ λ então
6              Crie um novo filho para r com rótulo t_i;
7      retorna BuscaEmProfundidade(r, λ);
8  fim
```

Algoritmo 4.1: Heurística para encontrar todos as cadeias válidas.

```
   Entrada: Nodo n de uma árvore de trincas, λ
   Saída: Conjunto de cadeias freqüentes representadas na subárvore de n
1  início
2      CadeiasFrequentes ← ∅ ;
3      para cada filho n_i de n faça
4          para cada irmão n_j de n_i cujo rótulo não seja menor que o de n_i faça
5              Crie um novo filho n_ij para n_i, com mesmo rótulo de n_j;
6              c ← a cadeia representada por n_ij;
7              se |M(c)| < λ então
8                  Remova o nodo n_ij da árvore;
9              senão
10                 Insira c em CadeiasFrequentes;
11         CadeiasFrequentes ← CadeiasFrequentes ∪ BuscaEmProfundidade(n_i);
12     retorna CadeiasFrequentes;
13 fim
```

Algoritmo 4.2: Busca em profundidade.

que será construída. A partir desses passos, as linhas 4 a 6, criam um filho para a raiz da árvore para cada trinca freqüente, ou seja, cujo conjunto de mapeamentos seja maior que λ. É interessante notar que, pelo mesmo princípio utilizado por algoritmos como o *Apriori*, para um grupo ser freqüente, é necessário que todos os seus elementos também o sejam. Até aqui, encontramos todas as cadeias freqüentes de tamanho um, que são representadas por apenas uma aresta. Então, a linha 7 chama o procedimento recursivo que realizará a busca em profundidade pela árvore, de forma a criar as cadeias de tamanho maior que um.

A busca em profundidade na árvore de trincas pode ser vista no Algoritmo 4.2. Os parâmetros de entrada são o próprio λ e um nodo que será a raiz da subárvore sendo tratada. O objetivo do procedimento é descobrir quais são os netos válidos da raiz atual

(note que somente os netos são gerados pois garante-se que um nodo seja uma raiz para a busca apenas se seus filhos já tiverem sido criados). A linha 4 indica que devemos expandir cada cadeia, representada por um filho n_i da raiz, com uma trinca que seja rótulo de um dos irmãos n_j de n_i. Lembramos que o rótulo de n_j não pode ser menor que o rótulo de n_i. Após criarmos um filho para n_i, podemos utilizar o Algoritmo 3.1 para checar se existem mapeamentos suficientes para que essa nova cadeia seja considerada freqüente (passo descrito nas linhas 5, 6 e 7). Na linha 8, garantimos que a árvore representará apenas cadeias válidas, removendo todos os nodos que simbolizem cadeias não freqüentes. A linha 10, por sua vez, garante a inclusão da cadeia c no grupo de cadeias freqüentes, caso ela o seja. A recursividade da função está presente na linha 11, em que uma nova busca em profundidade é iniciada, a partir de um filho da raiz da subárvore.

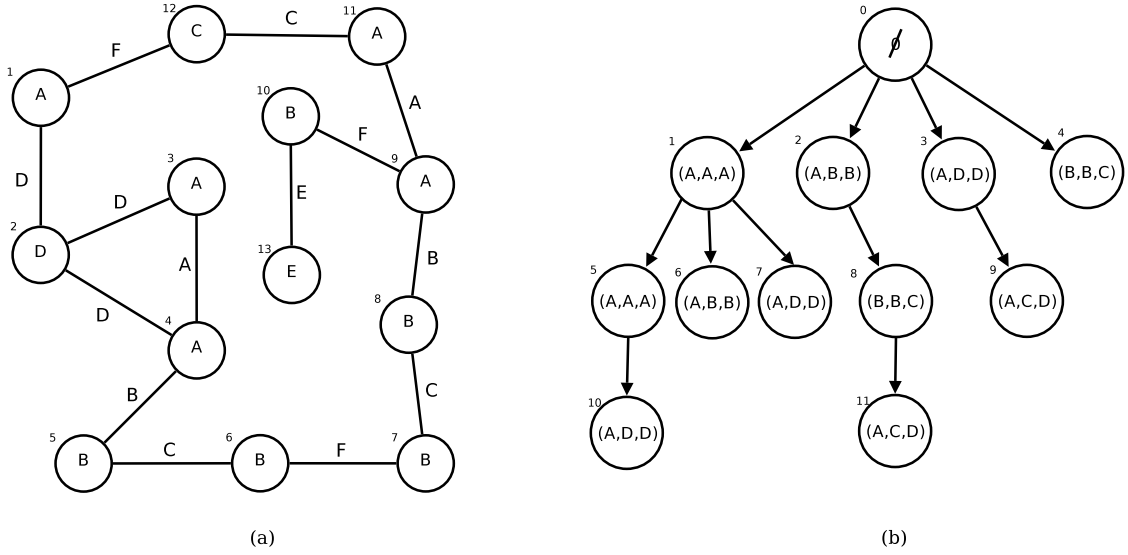

Figura 4.1. Exemplo de um caso de falha apresentado pela heurística.

Na heurística VL, conforme a definimos até aqui, o laço da linha 4 é o principal responsável por sua característica gulosa. Isso porque ela é baseada no raciocínio de algoritmos como o *Apriori*, onde uma tupla $\{a, b, c\}$ somente será freqüente se $\{a, b\}$, $\{b, c\}$ e $\{a, c\}$ também forem. Entretanto, esse fato não é válido para cadeias. Observe a Figura 4.1, onde temos um multigrafo e sua árvore de trincas (desenvolvida segundo os pseudo-códigos apresentados) e consideraremos um λ igual a dois. Sabemos que a cadeia $c = \{(A, A, A), (A, B, B), (B, B, C)\}$ é freqüente (pois $M(c) = \{\{(3, 4, A), (4, 5, B), (5, 6, C)\}, \{(9, 11, A), (8, 9, B), (7, 8, C)\}\}$), porém ela não será gerada pelo algoritmo na árvore, pois a cadeia $\{(A, A, A), (B, B, C)\}$ não é freqüente. Para que essa limitação não existisse, bastaria que a linha 4 tentasse criar um

novo filho para n_j a partir de todas as trincas freqüentes descobertas no passo inicial do Algoritmo 4.1. Todavia, essa opção tornaria a complexidade do algoritmo inviável, pois o número de trincas freqüentes é, em geral, muito grande.

4.2 Limitações da implementação

Da forma como definimos a heurística VL na seção anterior, seu processamento é ineficiente, devido à complexidade e à explosão de possibilidades. Vamos apresentar, agora, as principais adaptações realizadas na heurística apresentada no Algoritmo 4.1, com o objetivo de melhorar seu desempenho. Na última seção apresentamos os pseudocódigos que descrevem a heurística da Vizinhança Limitada após as adaptações.

4.2.1 Sobre o número de mapeamentos

Citamos que, para o processamento das linhas 5 a 7 do Algoritmo 4.2, poderíamos utilizar o Algoritmo 3.1. Entretanto, esse segundo algoritmo, da forma como apresentado, não é viável. Isso porque, para cada mapeamento da cadeia, analisa-se um a um os mapeamentos da trinca para a expansão. Como estamos trabalhando com grandes bases de dados, é perfeitamente normal ocorrer a expansão de uma cadeia com uma trinca, onde ambas possuem mil mapeamentos, o que exigiria um milhão de iterações. Esse custo é intratável, já que podem existir milhares de trincas distintas, e a operação de expansão é realizada para cada novo nodo gerado na árvore.

Para contornar esse problema, a heurística VL não armazenará todos os mapeamentos das cadeias da árvore. Definimos uma constante MAX e, com ela, a seguinte restrição:

> *Nenhum nodo da árvore de trincas deve possuir associado a si um número de mapeamentos maior que MAX.*

Sendo assim, se a cadeia que um nodo representa possuir um total de mapeamentos maior que MAX, mapeamentos devem ser descartados, até que a restrição seja atendida.

Para realizar esse descarte, optamos por selecionar mapeamentos de forma não determinista. Existem duas razões principais para isso: a escolha aleatória é extremamente simples, não embutindo complexidade nenhuma ao algoritmo; não existe nenhuma prova teórica que algum outro método obteria melhores resultados. Sendo assim, garantimos a restrição citada acima em dois momentos distintos, discutidos a seguir.

4.2.1.1 Descarte durante a avaliação de conjunto de mapeamentos

O principal momento no qual utilizamos a constante MAX, com o intuito de limitarmos o número de mapeamentos armazenados por nossa heurística para cada cadeia, é durante as avaliações do conjunto de mapeamentos (representadas pela linha 5 no Algoritmo 4.1 e pela linha 7 no Algoritmo 4.2). Originalmente, essas avaliações serviam apenas para evitar que cadeias inválidas continuassem na árvore. Entretanto, com o uso da nova constante, adicionamos uma nova verificação.

Suponha γ a cardinalidade do conjunto de mapeamentos sendo analisado e α uma variável aleatória uniforme no intervalo $(0, 1]$. Para todas as cadeias que são válidas, calculamos, agora, o número d de mapeamentos que extrapolam o limite dado por MAX:

$$d = \gamma - MAX$$

Para os casos em que $d > 0$, removemos d mapeamentos distintos da cadeia. Esses mapeamentos são selecionados de forma não determinista, sendo seus índices definidos pela seguinte fórmula:

$$\lceil \alpha \cdot \gamma \rceil.$$

Apenas como esclarecimento, chamamos de índice de um mapeamento apenas um número associado a cada um dos mapeamentos, de forma a identificar os elementos do conjunto de forma única.

```
   Entrada: c_n, M(c_n)
1  início
2      retirar ← |M(c_n)| − MAX;
3      IndicesDistintos ← ∅;
4      enquanto retirar > 0 faça
5          α ← geraAleatorio();
6          Insere ⌈α · |M(c_n)|⌉ em IndicesDistintos;
7          retirar = retirar − 1;
8      para cada i ∈ IndicesDistintos faça
9          Retire o mapeamento de índice i do conjunto M(c_n);
10 fim
```

Algoritmo 4.3: Procedimento para eliminar mapeamentos aleatoriamente.

Assim, podemos definir um procedimento de remoção dos mapeamentos excedentes, sendo apresentado no Algoritmo 4.3. Esse procedimento recebe como parâmetros uma dada cadeia e seus mapeamentos e, caso existam mais mapeamentos que o permitido por MAX, remove do conjunto a quantidade necessária de elementos para atender a

restrição. Essa remoção feita pode ser avaliada como um processo de enumeração não-determinista dos mapeamentos da cadeia, escolhendo-se, dentro de um limite, quais mapeamentos são mantidos. É interessante notar que a função *geraAleatório()* tem exatamente o mesmo comportamento que uma variável aleatória uniforme no intervalo $(0, 1]$.

4.2.1.2 Descarte durante geração de candidatos

O outro ponto importante para controlar o número de mapeamentos mantidos para cada cadeia é durante a geração dos mapeamentos de uma nova cadeia, mostrada no Algoritmo 3.1. Com a definição da constante MAX, o pior caso (uma cadeia com MAX mapeamentos e uma trinca de MAX instâncias) causaria MAX^2 iterações para encontrarmos os mapeamentos da expansão. Entretanto, também seria possível encontrarmos MAX^2 mapeamentos válidos, o que obrigaria a retirar uma quantidade excessiva de mapeamentos ($MAX^2 - MAX$, que é $O(MAX^2)$).

A idéia é, então, controlar o número de iterações realizado pelos laços do procedimento em questão. Para tal, criamos outra constante, denominada $FATOR$, que sempre assumirá algum valor pequeno (entre 2 e 5, por exemplo). Usaremos o valor dessa constante para garantir que os os laços citados produzam $FATOR \cdot MAX$ iterações no pior caso e, portanto, fazendo com que o procedimento possua complexidade linear $O(MAX)$. Cabe ressaltar que $FATOR$ deverá ser sempre maior que 1, por um motivo simples: nem todas as iterações constroem um mapeamento válido para a nova cadeia. Em outras palavras, precisamos de uma "margem de segurança", para que o total de iterações produza MAX elementos válidos.

De acordo com o Algoritmo 3.1, o total de iterações T inicialmente seria dado por:

$$T = |M(c_n)| \cdot |I(t_i)|$$

Assim, caso $T > MAX \cdot FATOR$, iremos desprezar algumas iterações, com o intuito de diminuir o total de mapeamentos gerados. O número R de iterações que serão evitadas é:

$$R = T - MAX \cdot FATOR$$

Dessa forma, para que o laço tenha a complexidade desejada, a probabilidade ϕ de não executarmos uma dada iteração dos laços é dada por:

$$\phi = \frac{R}{T}$$

O Algoritmo 4.4 apresenta a versão modificada do Algoritmo 3.1, que calcula o conjunto de mapeamentos de uma expansão utilizando aleatoriedade. A função

```
   Entrada: c_n, M(c_n), t_i, I(t_i)
   Saída: M(c')
 1 início
 2     iteracoes ← M(c_n) · I(t_i);
 3     retirar ← iteracoes − (FATOR · MAX);
 4     corte ← retirar / iteracoes;
 5     M(c') ← ∅;
 6     para cada m_k ∈ M(c_n) faça
 7         se retirar > MAX então
 8             α ← geraAleatorio();
 9         senão
10             α ← 1 ;
11         se  α > corte/3 então
12             para cada i_q ∈ I(t_i) faça
13                 se retirar > 0 então
14                     α ← geraAleatorio();
15                 senão
16                     α ← 1 ;
17                 se  α > corte então
18                     se i_q ∉ m_k então
19                         se m_k possui um dos dois índices de nodo de i_q então
20                             e ← m_k ∪ {i_q};
21                             se e ∉ M(c') então
22                                 Insira e no conjunto M(c');
23                 senão
24                     retirar = retirar − 1;
25         senão
26             retirar = retirar − |I(t_i)|;
27     retorna M(c');
28 fim
```

Algoritmo 4.4: Novo procedimento para encontrar mapeamentos de uma expansão.

geraAleatório() novamente representa uma variável aleatória uniforme no intervalo $(0, 1]$. Além de utilizarmos a probabilidade ϕ para eliminar a geração de candidatos no laço mais interno (linha 17), utilizamos uma probabilidade menor que ϕ (no caso, escolhemos $\frac{\phi}{3}$ por simplicidade) para evitar iterações do laço externo (linha 11). Retirar iterações do laço externo possui maior impacto na redução de complexidade, entretanto, a retirada de cada uma delas deve ser feita de forma cuidadosa, para não eliminar iterações em excesso. Isso justifica o valor $\frac{\phi}{3}$ e também a verificação feita na linha 7, que permite apenas evitar uma iteração do laço externo se ainda existirem mais de MAX mapeamentos a serem removidos, pois o laço interno pode possuir até MAX iterações. As linhas 24 e 26 são utilizadas para, justamente, manter atualizado esse número de mapeamentos a serem eliminados. Lembramos que as condições apresentadas nas linhas 18 e 21 não são desconsideradas para a aplicação utilizada nesse trabalho, pois grupos de fóruns não possuirão elementos repetidos.

É fundamental notar que a restrição imposta pela constante MAX faz com que a heurística deixe de ser determinista. Isso porque, dependendo da retirada de mapeamentos feita, duas saídas podem ser muito distintas para um mesmo multigrafo e parâmetro de entrada.

4.2.2 Sobre a profundidade máxima da árvore

Como a heurística VL é voltada para minerar grandes bases de dados, outra limitação importante a ser citada envolve a condição de parada da recursividade inerente à busca em profundidade na árvore de trincas. Da maneira como apresentamos o Algoritmo 4.2, temos a garantia que uma execução da busca termina pois, para todos as recursões possíveis, em algum momento a raiz da subárvore não terá filhos. Entretanto, essa condição pode acontecer somente para profundidades muito grandes, o que tornaria a heurística inviável em termos de espaço de memória necessário, pois o número de nodos da árvore tem um crescimento exponencial em relação a profundidade da busca e precisamos armazenar os mapeamentos das cadeias representadas por todos os nodos.

Sendo assim, criamos um novo parâmetro p para controlar a busca em profundidade, que indica o maior tamanho possível de cadeias a serem encontradas. Com isso, antes de explorar os filhos de uma raiz de subárvore, a busca em profundidade analisa a profundidade dessa raiz. Se os netos da mesma (a serem gerados) possuírem profundidade maior que p, o procedimento retornará sem realizar ação alguma. Podemos ver essa alteração na linha 4 do Algoritmo 4.6.

Essa mudança afeta os resultados encontrados pela heurística VL, justamente por impedir que a busca em profundidade encontre padrões maiores que p. Contudo, conforme veremos nos experimentos do próximo capítulo, alguns valores de p permitem

a geração de resultados muito significativos.

4.2.3 A nova versão da heurística

As mudanças citadas nas subseções anteriores possuem grandes impactos na heurística VL, conforme definida na primeira seção desse capítulo. Assim, apresentamos agora as novas versões, que refletem essas mudanças.

Primeiramente, apresentamos o Algoritmo 4.5 em substituição ao Algoritmo 4.1, que resume a própria heurística VL. São três as mudanças: a existência de uma terceira entrada, p, que define a profundidade máxima a ser explorada na árvore de trincas; a linha 6, que é uma chamada ao procedimento apresentado no Algoritmo 4.3; e a linha 8, que executa a busca em profundidade modificada, a ser apresentada a seguir. Lembramos que MAX e $FATOR$ são consideradas constantes, portanto não as tratamos como parâmetros de entrada e/ou saída em nenhum dos algoritmos apresentados.

```
   Entrada: λ, m_g, p
   Saída: Cadeias de trincas freqüentes
1  início
2  |  T ← EncontraTrincas(m_g);
3  |  Inicialize a árvore de trincas com a raiz r;
4  |  para cada trinca t_i ∈ T faça
5  |  |  se |M({t_i})| ≥ λ então
6  |  |  |  EliminaMapeamentos(t_i,M({t_i}));
7  |  |  |  Crie um novo filho para r com rótulo t_i;
8  |  retorna BuscaEmProfundidade(r, λ, p);
9  fim
```

Algoritmo 4.5: Nova versão da Heurística, com adaptações embutidas.

Resta definir o Algoritmo 4.6, que altera o comportamento da busca em profundidade apresentado no Algoritmo 4.2. O parâmetro de profundidade máxima, p, também foi adicionado aqui. A linha 4 representa a inclusão da nova condição de parada, limitando o tamanho máximo dos subgrafos a serem descobertos. O processo de criação da nova cadeia e seus mapeamentos, representado pelas linhas 7 e 8, é realizado através do Algoritmo 4.4. A linha 12 executa o Algoritmo 4.3. Por fim, a linha 13 representa a recursividade do procedimento.

Apenas como um exemplo, vamos descrever uma execução do algoritmo a partir do multigrafo da Figura 3.9, utilizando $\lambda = 2$. Para facilitar o entendimento, a profundidade máxima e a constante MAX estarão definidas como: $p = MAX = \infty$, ou seja, ambas não afetam o comportamento da heurística VL. Inicialmente, todo o primeiro nível da árvore de cadeias, apresentada na Figura 3.10, é gerado. Entretanto, os nodos

```
   Entrada: Nodo n de uma árvore de trincas, λ, p
   Saída: Conjunto de cadeias frequentes representadas na subárvore de n
 1 início
 2     CadeiasFrequentes ← ∅ ;
 3     aux ← profundidade do nodo n;
 4     se aux + 2 ≤ p então
 5         para cada filho n_i de n faça
 6             para cada irmão n_j de n_i cujo rótulo não seja menor que n_i faça
 7                 Crie um novo filho n_ij para n_i, com mesmo rótulo de n_j;
 8                 c ← a cadeia representada por n_ij;
 9                 se |M(c)| < λ então
10                     Remova o nodo n_ij da árvore;
11                 senão
12                     EliminaMapeamentos(c,M(c));
13                     Insira c em CadeiasFrequentes;
14             CadeiasFrequentes ← CadeiasFrequentes ∪
               BuscaEmProfundidade(n_i);
15     retorna CadeiasFrequentes;
16 fim
```

Algoritmo 4.6: Novo procedimento para realizar a busca em profundidade.

1 e 3 são eliminados, por não serem freqüentes. A busca em profundidade, iniciada a partir do nodo 0, gera os nodos 8 e 9 como filhos do nodo 2. A recursão desse procedimento, então, cria o nodo 13. A recursão então pára, pois não existem mais alternativas de expansão das cadeias. Voltando o controle da busca para o nodo 0, o nodo 11 é gerado. Esse controle é passado para o nodo 4, mas nenhuma expansão é possível. Ao retornar o controle da busca em profundidade para o nodo 0, percebe-se que o nodo 5 não gera filhos e, assim, a execução termina. Basta agora identificar as cadeias de todos os nodos restantes na árvore, pois todas representam padrões freqüentes.

A partir desse momento, temos uma heurística não-determinista que, dado um multigrafo, consegue minerar padrões freqüentes. A última versão da heurística VL apresentada foi implementada utilizando-se a linguagem de programação C. Porém, antes de descrevermos os experimentos realizados, iremos, a seguir, demonstrar o seu funcionamento através de um exemplo prático.

4.3 Um exemplo de funcionamento

Para ilustrar a aplicação da heurística descrita, iremos agora apresentar um exemplo de funcionamento da mesma, onde definiremos um conjunto de dados de entrada para,

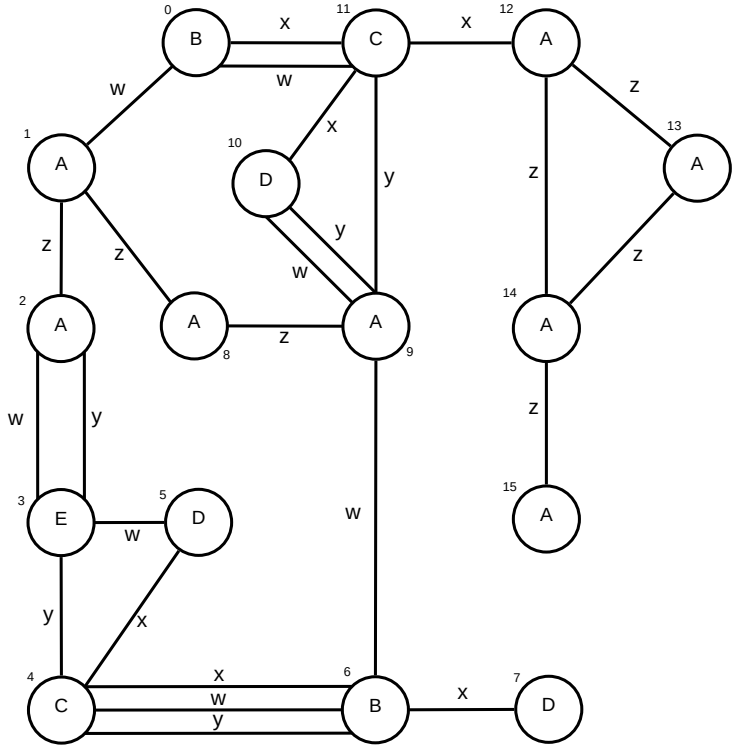

Figura 4.2. Multigrafo usado para exemplo de execução da heurística.

então, mostrarmos os passos do algoritmo e os resultados obtidos.

Conforme o conceito da Seção 3.1, para definirmos nosso multigrafo, precisamos dos conjuntos R, V, A, l_v, l_a, definidos a seguir:

- $R = R' \cup R''$, onde $R' = \{A, B, C, D, E\}$ e $R'' = \{x, y, z, w\}$
- $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
- $A = \{(0, 1), (0, 11), (1, 2), (1, 8), (2, 3), (3, 4), (3, 5), (4, 5), (4, 6), (6, 7), (6, 9), (8, 9), (9, 10), (9, 11), (10, 11), (11, 12), (12, 13), (12, 14), (13, 14), (14, 15)\}$
- $l_v = \{(0, B), (1, A), (2, A), (3, E), (4, C), (5, D), (6, B), (7, D), (8, A), (9, A), (10, D), (11, C), (12, A), (13, A), (14, A), (15, A)\}$
- $l_a = \{(0, 1, w), (0, 11, w), (0, 11, x), (1, 2, z), (1, 8, z), (2, 3, w), (2, 3, y), (3, 4, y), (3, 5, w), (4, 5, x), (4, 6, w), (4, 6, x), (4, 6, y), (6, 7, x), (6, 9, w), (8, 9, z), (9, 10, w), (9, 10, y), (9, 11, y), (10, 11, x), (11, 12, x), (12, 13, z), (12, 14, z), (13, 14, z), (14, 15, z)\}$

Com esses conjuntos, temos o multigrafo $g = (V, A, l_v, l_a)$ apresentado na Figura 4.2. Apenas por ilustração, poderiamos pensar nesse grafo como uma representação de redes de co-autoria, onde cada autor estaria representado por um nodo e classificado por

t	$I(t)$
(A, A, z)	$\{(1, 2, z), (1, 8, z), (8, 9, z), (12, 13, z), (12, 14, z), (13, 14, z), (14, 15, z)\}$
(A, B, w)	$\{(0, 1, w), (6, 9, w)\}$
(A, C, x)	$\{(11, 12, x)\}$
(A, C, y)	$\{(9, 11, y)\}$
(A, D, w)	$\{(9, 10, w)\}$
(A, D, y)	$\{(9, 10, y)\}$
(A, E, w)	$\{(2, 3, w)\}$
(A, E, y)	$\{(2, 3, y)\}$
(B, C, w)	$\{(0, 11, w), (4, 6, w)\}$
(B, C, x)	$\{(0, 11, x), (4, 6, x)\}$
(B, C, y)	$\{(4, 6, y)\}$
(B, D, x)	$\{(6, 7, x)\}$
(C, D, x)	$\{(4, 5, x), (10, 11, x)\}$
(C, E, y)	$\{(3, 4, y)\}$
(D, E, w)	$\{(3, 5, w)\}$

Tabela 4.1. Trincas do multigrafo da Figura 4.2 e suas instâncias.

sua principal área de pesquisa. As arestas poderiam indicar co-autoria de um artigo, sendo classificadas pela área da conferência no qual o artigo foi publicado. Os padrões resultados de nosso algoritmo nos indicariam, então, como os autores interagem entre si, trabalhando em diferentes linhas de pesquisa.

Para que possamos aplicar a heurística apresentada nesta dissertação, entretanto, ainda precisamos de algumas definições auxiliares, como:

- a ordenação entre os elementos de R: $A < B < C < D < E < w < x < y < z$
- as constates $\lambda = 2$, $p = 3$, $MAX = 6$ e $FATOR = 4$[1]

Com esses dados, iniciaremos então a execução da nossa heurística sobre o multigrafo.

O primeiro passo do algoritmo é a identificação das trincas e suas instâncias. No caso do multigrafo desse exemplo, teriamos os dados da Tabela 4.1. Com tal conjunto de instâncias, o algoritmo constrói o primeiro nível da árvore de trincas (com cadeias de tamanho igual a um), conforme podemos visualizar na Figura 4.3(a). Os dados desse nível da árvore podem ser vistos na Tabela 4.2. Note que todas as cadeias com menos de λ mapeamentos foram descartadas. Além disso, um dos mapeamentos da cadeia (A, A, z) também foi descartado, devido a constante MAX. A escolha aleatória foi simulada ao eliminarmos a aresta $(13, 14, z)$.

[1]Esse valor para $FATOR$ foi escolhido apenas para facilitar a compreensão do exemplo.

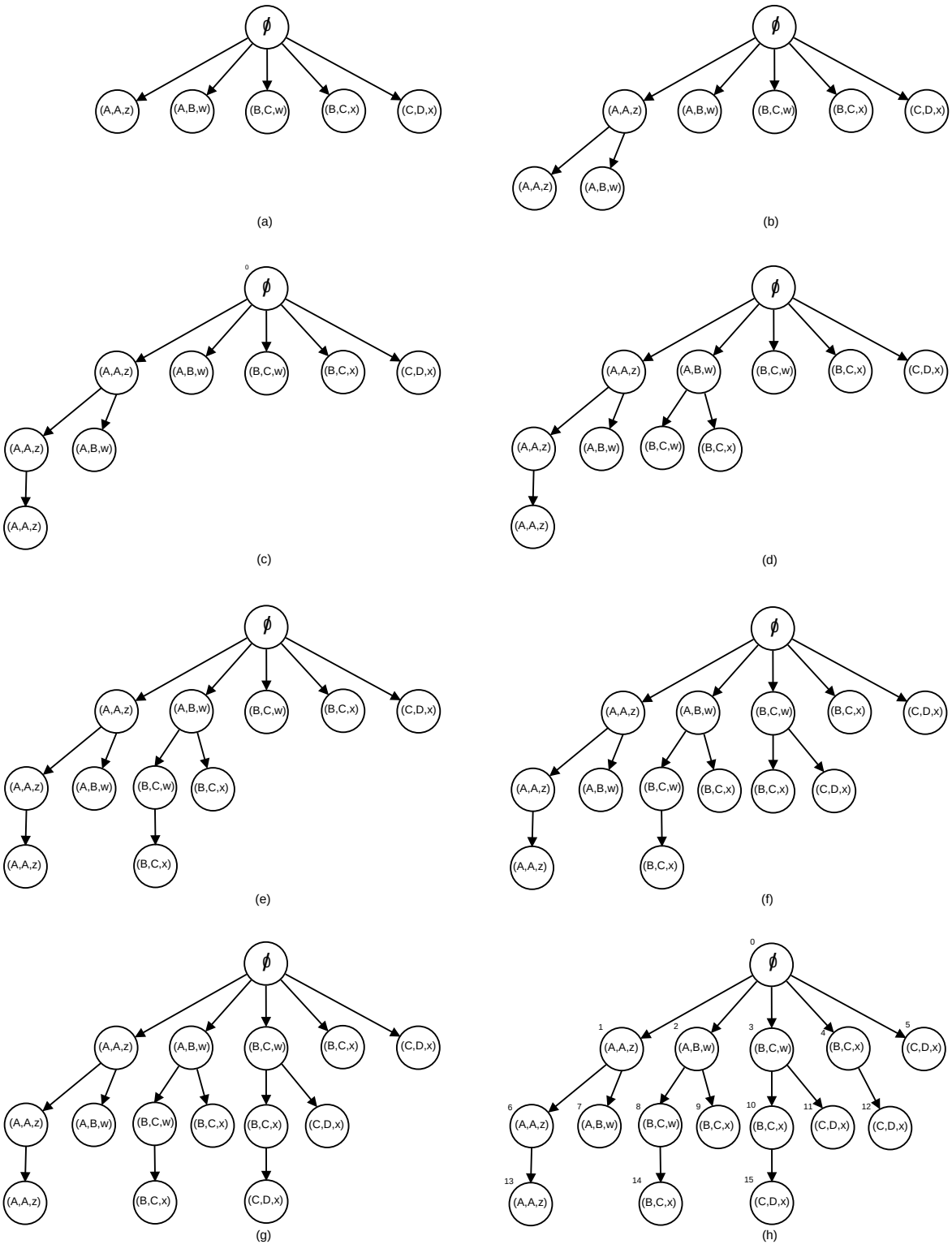

Figura 4.3. Formação da árvore de trincas para o multigrafo sendo processado.

Agora, o processo de expansão das cadeias tem início. Usando a busca em profundidade, a primeira cadeia a ser expandida é a $\{(A, A, z)\}$. Essa expansão pode ser vista

c	$M(c)$
$\{(A, A, z)\}$	$\{\{(1, 2, z)\}, \{(1, 8, z)\}, \{(8, 9, z)\}, \{(12, 13, z)\}, \{(12, 14, z)\}, \{(14, 15, z)\}\}$
$\{(A, B, w)\}$	$\{\{(0, 1, w)\}, \{(6, 9, w)\}\}$
$\{(B, C, w)\}$	$\{\{(0, 11, w)\}, \{(4, 6, w)\}\}$
$\{(B, C, x)\}$	$\{\{(0, 11, x)\}, \{(4, 6, x)\}\}$
$\{(C, D, x)\}$	$\{\{(4, 5, x)\}, \{(10, 11, x)\}\}$

Tabela 4.2. Cadeias de tamanho um selecionadas e seus mapeamentos.

c	$M(c)$
$\{(A, A, z), (A, A, z)\}$	$\{\{(1, 2, z), (1, 8, z)\}, \{(1, 8, z), (8, 9, z)\}, \{(12, 13, z), (12, 14, z)\}, \{(12, 14, z), (14, 15, z)\}\}$
$\{(A, A, z), (A, B, w)\}$	$\{\{(1, 2, z), (0, 1, w)\}, \{(1, 2, z), (6, 9, w)\}\}$

Tabela 4.3. Cadeias geradas pela expansão de $\{(A, A, z)\}$ e seus mapeamentos.

c	$M(c)$
$\{(A, A, z), (A, A, z), (A, A, z)\}$	$\{\{(1, 2, z), (1, 8, z), (8, 9, z)\}, \{(12, 13, z), (12, 14, z), (14, 15, z)\}\}$

Tabela 4.4. Cadeia gerada pela expansão de $\{(A, A, z), (A, A, z)\}$ e seus mapeamentos.

c	$M(c)$
$\{(A, B, w), (B, C, w)\}$	$\{\{(0, 1, w), (0, 11, w)\}, \{(6, 9, w), (4, 6, w)\}\}$
$\{(A, B, w), (B, C, x)\}$	$\{\{(0, 1, w), (0, 11, x)\}, \{(6, 9, w), (4, 6, x)\}\}$

Tabela 4.5. Cadeias geradas pela expansão de $\{(A, A, z)\}$ e seus mapeamentos.

na Figura 4.3(b). Os mapeamentos das novas cadeias são apresentados na Tabela 4.3.

A próxima cadeia a ser expandida é $\{(A, A, z), (A, A, z)\}$. O resultado pode ser visto na Figura 4.3(c) e os mapeamentos são apresentados na Tabela 4.4.

Note que agora temos uma cadeia com três trincas. Devido a constante p (definida com valor 3) nesse exemplo, não tentaremos expandir a cadeia $\{(A, A, z), (A, A, z), (A, A, z)\}$. O próximo passo será, então, a expansão da cadeia $\{(A, A, z), (A, B, w)\}$. Entretanto, a única cadeia possível ($\{(A, A, z), (A, B, w), (A, B, w)\}$) não possui λ mapeamentos. Assim, a busca retrocede um nível e tenta a expansão de $\{(A, B, w)\}$. As cadeias criadas nesse passo podem ser vistas na Figura 4.3(d), sendo os seus mapeamentos apresentados na Tabela 4.5.

A próxima cadeia a ser expandida é $\{(A, B, w), (B, C, w)\}$, alterando a árvore de

c	$M(c)$
$\{(A,B,w),(B,C,w),(B,C,x)\}$	$\{\{(0,1,w),(0,11,w),(0,11,x)\}$ $\{(6,9,w),(4,6,w),(4,6,x)\}\}$

Tabela 4.6. Cadeia gerada pela expansão de $\{(A,B,w),(B,C,w)\}$ e seus mapeamentos.

c	$M(c)$
$\{(B,C,w),(B,C,x)\}$	$\{\{(0,11,w),(0,11,x)\},\{(4,6,w),(4,6,x)\}\}$
$\{(B,C,w),(C,D,x)\}$	$\{\{(0,11,w),(10,11,x)\},\{(4,6,w),(4,5,x)\}\}$

Tabela 4.7. Cadeias geradas pela expansão de $\{(B,C,w)\}$ e seus mapeamentos.

c	$M(c)$
$\{(B,C,w),(B,C,x),(C,D,x)\}$	$\{\{(0,1,w),(0,11,x),(10,11,x)\}$ $\{(4,6,w),(4,6,x),(4,5,x)\}\}$

Tabela 4.8. Cadeia gerada pela expansão de $\{(B,C,w),(B,C,x)\}$ e seus mapeamentos.

c	$M(c)$
$\{(B,C,x),(C,D,x)\}$	$\{\{(0,11,x),(10,11,x)\},\{(4,6,x),(4,5,x)\}\}$

Tabela 4.9. Cadeia gerada pela expansão de $\{(B,C,x)\}$ e seus mapeamentos.

trincas conforme a Figura 4.3(e). Novamente, os mapeamentos estão apresentados na Tabela 4.6.

Não tentamos expandir $\{(A,B,w),(B,C,w),(B,C,x)\}$ novamente pela definição da constante p. A expansão de $\{(A,B,w),(B,C,x)\}$ não gera cadeias válidas. Logo, a próxima expansão será a da cadeia $\{(B,C,w)\}$, cujos mapemantos estão na Tabela 4.7. A alteração na árvore de trincas pode ser vista na Figura 4.3(f).

Seguindo a busca em profundidade, o próximo passo é a expansão de $\{(B,C,w),(B,C,x)\}$, que pode ser observada pela Figura 4.3(g) e pela Tabela 4.8. Como a cadeia resultante possui três trincas, ela não expandida. A tentativa de expansão da cadeia $\{(B,C,w),(C,D,x)\}$ não gera candidatos aceitáveis. Forçando a busca a retroceder para o nível anterior da árvore.

Expandimos agora a cadeia $\{(B,C,x)\}$, alterando a árvore para sua versão final, vista na Figura 4.3(h). Os mapeamentos resultantes dessa operação estão na Tabela 4.9. A expansão de $\{(B,C,x),(C,D,x)\}$ não resulta em trincas válidas, assim como a expansão $\{(C,D,x)\}$. Sendo assim, o processamento do algoritmo termina.

O resultado da execução é dado pela união das cadeias e respectivos mapeamentos das tabelas 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 e 4.9. Esse conjunto de cadeias de trincas freqüentes nos indicaria, para esse exemplo, quais parcerias os autores procuram realizar quando fazem publicações em áreas que não são sua especialidade. Com tais resultados, poderiam ser realizados estudos para explorar a iteração entre autores de forma a promover novos avanços científicos.

Terminado o nosso exemplo de execução da heurística, o trabalho desse capítulo termina. No capítulo a seguir, descrevemos uma série de experimentos realizados com tal implementação, que nos permitirão realizar análises importantes sobre o comportamento empírico de nossa técnica.

Capítulo 5

Avaliação Experimental

Neste capítulo, vamos descrever e analisar os experimentos realizados com a implementação de heurística definida no capítulo anterior. Assim, detalharemos as bases de dados, a metodologia que utilizamos, e apresentaremos os resultados obtidos para, finalmente, discutirmos sobre a eficiência e a qualidade da técnica.

5.1 Bases de dados

Os experimentos apresentados neste capítulo foram realizados sobre três redes de co-autoria, construídas a partir das publicações provenientes de trinta e três faculdades brasileiras e vinte e duas internacionais, ao longo de mais de setenta anos. A Tabela 5.1 apresenta algumas informações dessa massa de dados, que abrange mais de trezentos mil artigos e cem mil autores. A partir de cada rede, geramos uma base apta a ser utilizada como uma entrada válida da heurística VL.

A primeira base de dados, que denominamos Base Completa, possui informações referentes às publicações de 2.008 professores da área de Computação de 8 faculdades nacionais e 22 internacionais, assim como informações referentes aos artigos dos co-

	Base Completa	Base Resumida	Base Pós
Autores	176.537	35.201	6.414
Artigos	352.766	55.898	6.392
Conferências	6124	3.440	1.429
Faculdades Nacionais	8	8	33
Faculdades Internacionais	22	22	0
Período Analisado	De 1936 a 2008	De 1954 a 2007	De 1971 a 2007
Fóruns classificados	Não	Sim	Não

Tabela 5.1. Informações relativas às redes de co-autoria.

autores e de seus respectivos co-autores, ou seja, explorando 3 níveis de co-autoria. Ela é a maior das três bases e é utilizada principalmente em testes, para analisarmos o desempenho da heurística frente à variação de alguns dos diversos parâmetros envolvidos.

A segunda, chamada Base Resumida, é construída com os mesmos dados da Base Completa, com duas diferenças: não estão inseridos os artigos do terceiro nível de co-autoria e todos os fóruns foram manualmente classificados em 32 áreas de conhecimento distintas. Na Tabela 5.2, apresentamos quais são essas áreas e quantos artigos existem em cada uma delas. O campo ID da tabela será utilizado como um identificador, para distinguir as áreas no restante da nossa avaliação. Essa base de dados é utilizada nos experimentos que analisam a qualidade dos grupos obtidos pela heurística.

ID	Área	Artigos
0	Sem classificação	62
1	Algoritmos e Teoria da Computação	193
2	Aprendizado de Máquina	116
3	Arquitetura de Computadores, Processamento de Alto Desempenho	172
4	Sistemas Operacionais	16
5	Bancos de Dados, Bibliotecas Digitais	267
6	Recuperação de Informação	36
7	Mineração de Dados	31
8	Biologia Computacional	61
9	Computação Aplicada	66
10	Computação Gráfica, Processamento de Imagens	171
11	Visão Computacional	64
12	Computação Ubíqua	54
13	Concepção de Circuitos Integrados	69
14	Engenharia de Software, Métodos Formais	176
15	Formalismos, Lógica, Semântica da Computação	161
16	Geo-informática	26
17	Informática na Educação	25
18	Inteligência Artificial	228
19	Interação Humano Computador, Sistemas Colaborativos	83
20	Jogos e Entretenimento, Realidade Virtual	25
21	Linguagens de Programação	200
22	Pesquisa Operacional e Otimização Combinatória	50
23	Processamento de Língua Natural	40
24	Redes de Computadores, Sistemas Distribuídos, Sistemas P2P	235
25	Robótica, Controle e Automação	32
26	Segurança e Privacidade	98
27	Simulação e Modelagem	47
28	Sistemas de Informação	62
29	Sistemas Embarcados, Sistemas de Tempo Real, Sistemas Tolerantes a Falhas	59
30	Web, Sistemas Multimídia e Hipermídia	139
31	Multi-temáticas	376

Tabela 5.2. Informações das áreas presentes na Base Pós.

Já a terceira base de dados, a Base Pós, apresenta informações de 33 programas de pós-graduação, todos brasileiros. Ao todo, possui como autores 940 professores e seus 5440 co-autores. Por ser a menor das três e também não possuir as conferências classificadas em áreas, essa base é utilizada nos experimentos que analisam desempenho, como um contraponto à primeira base.

A seguir, iremos descrever como esses dados foram manipulados para gerar três multigrafos, que serão a entrada da heurística VL. Detalhamos, também, como modelar os dados como conjuntos de transações, possibilitando o uso do algoritmo *Apriori* para encontrar grupos freqüentes de conferências (cujos resultados serão comparados com os obtidos pela heurística). Esse segundo passo é importante, pois utilizamos a comparação entre os resultados de ambas as técnicas para avaliar o potencial de nosso trabalho.

5.1.1 Modelagem baseada em multigrafo

Para que fosse possível utilizar a heurística desenvolvida, foi preciso gerar multigrafos a partir das redes de co-autoria citadas. Como o objetivo do trabalho é classificar conferências em áreas do conhecimento, desconsiderando informações específicas dos autores, geramos um multigrafo, a partir de cada base de dados, da seguinte maneira:

- criamos um nodo para cada autor, sendo que todos os nodos possuem o mesmo rótulo. Os autores não são diferenciados porque o agrupamento de fóruns não distingue autores, sendo que diferenciá-los iria atrapalhar o processo.
- se dois autores forem co-autores de um (ou mais) artigo(s) em uma dada conferência c, existirá uma aresta ligando os nodos que representam esses autores. Essa aresta possuirá como rótulo o nome de c. Assim, ao agruparmos padrões semelhantes, encontramos relações de co-autoria comuns.

É interessante notar que, com essas condições, o multigrafo gerado não será, necessariamente, conexo. Isso, entretanto, não é problema, pois a heurística não assume tal restrição (em outras palavras, nossa técnica aceita como entrada uma floresta de multigrafos conexos).

Dessa forma, geramos três multigrafos (um para cada rede apresentada na Tabela 5.1), cujos dados podem ser vistos na Tabela 5.3. Um dado interessante dos multigrafos é o que chamamos de *multiplicidade* média, que representa o número médio de arestas entre dois nodos conectados (se fosse um grafo simples, seu valor seria um). Os valores encontrados (próximos a $1,5$ em todas as bases), indicam que o número de relacionamentos múltiplos nos multigrafos é significativo. Ressaltamos ainda o impressionante número de arestas da maior base (acima de um milhão e duzentos mil).

Na Figura 5.1, apresentamos um gráfico com a distribuição cumulativa de arestas por conferência para a Base Completa, onde podemos perceber que mais da metade do total de conferências possui menos de cinqüenta arestas no respectivo multigrafo. Em contrapartida, apenas três conferências possuem acima de dez mil arestas, sendo elas: *Nucleic Acids Research* (com 22.575 arestas), *Supercomputing* (com 12.745) e *Bionformatics* (com 10.078). Outro segmento importante refere-se às conferências com mais de cem e menos de trezentas arestas, que representam cerca de 30% do total. Esse valor é útil para validarmos os valores de suporte mínimo utilizados nos experimentos, mais adiante.

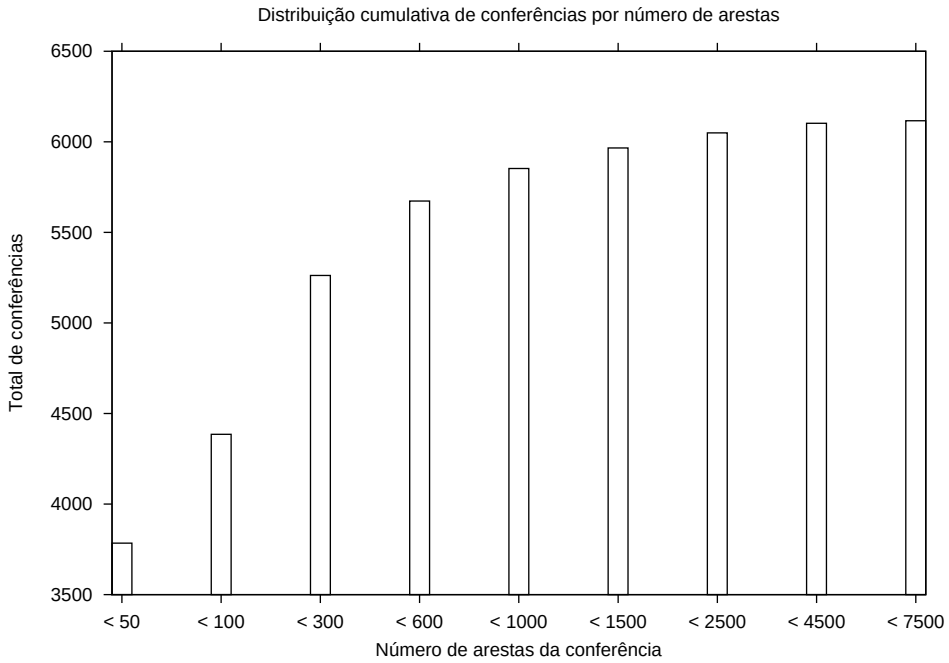

Figura 5.1. Distribuição cumulativa de fóruns para a Base Completa.

Ao utilizarmos a heurística VL em uma dessas bases, encontraremos uma série de padrões. Para entendermos esses padrões, lembramos os conceitos estabelecidos no capítulo anterior. Como os nodos possuem o mesmo rótulo, as trincas serão diferenciadas apenas pelo nome das conferências. Assim, uma cadeia encontrada representa um

	Base Completa	Base Resumida	Base Pós
Nodos	176.537	35.201	6.414
Arestas	1.201.218	216.180	28023
Multiplicidade média	1,59	1,55	1,49

Tabela 5.3. Dados dos multigrafos gerados.

conjunto de conferências correlacionadas. Resta, agora, apenas descrevermos brevemente o que os padrões freqüentes encontrados. Podemos destacar o seguinte:

> Cada padrão freqüente representa um grupo distinto de conferências que aparecem conectadas diversas vezes no multigrafo.

As conferências que constam em um padrão são consideradas correlatas pelo fato do mesmo conjunto de autores ter publicado nas mesmas. Logo, um padrão freqüente significa que muitos autores relacionam seus trabalhos em uma malha que envolve as conferências desse padrão, permitindo, assim, a classificação desse conjunto de conferências em uma Área (ou sub-Área).

5.1.2 Modelagem baseada em conjunto de transações

Nesta subseção, descrevemos como gerar uma entrada válida para o algoritmo *Apriori*, a partir da rede de co-autoria. O *Apriori* é um algoritmo que encontra, a partir de uma coleção de transações, conjuntos de itens freqüentes correlacionados entre si, isto é, conjuntos de itens que co-ocorrem em um número mínimo de transações. Como queremos agrupar conferências, iremos gerar uma coleção de transações da seguinte forma:

- uma transação por autor, representada por uma linha no arquivo de entrada sendo gerado.
- cada transação possui como itens as conferências nas quais esse autor possui pelo menos uma publicação (como autor principal ou co-autor).

Na Tabela 5.4, apresentamos um exemplo simples de uma coleção de transações gerada a partir de uma rede de co-autoria qualquer. Note que cada linha representa um autor distinto.

ConferênciaA ConferênciaB ConferênciaC
ConferênciaA ConferênciaC Conferência D Conferência F
ConferênciaB
ConferênciaC ConferênciaE

Tabela 5.4. Exemplo de um conjunto de transações.

Os resultados obtidos serão um conjunto de conferências que co-ocorrem diversas vezes nas transações da coleção, ou seja, conferências onde muitos autores concentram suas publicações. Embora cada padrão freqüente também represente uma Área (ou

sub-Área), apenas conferências que possuírem uma relação direta e explícita (muitos autores que publicam necessariamente em todas) serão agrupadas, o que nem sempre é o desejado.

5.2 Resultados experimentais

Nesta seção, apresentamos e avaliamos os principais resultados obtidos a partir dos experimentos realizados. Assim, na subseção 5.2.1, definimos quais estratégias de testes foram desenvolvidas para analisar a heurística VL. Em seguida, dividimos a apresentação dos resultados de acordo com tais estratégias.

5.2.1 Metodologia experimental

Os experimentos realizados neste trabalho visaram avaliar dois pontos principais:

1. o impacto dos diversos parâmetros no desempenho da heurística (em termos de tempo de execução e quantidade de padrões encontrados).

2. a qualidade dos agrupamentos gerados, através do estudo do comportamento não determinista nos resultados e da comparação com resultados provenientes do algoritmo *Apriori*.

Dessa forma, nas duas próximas subseções, descrevemos separadamente cada ponto citado acima.

Para os testes, utilizamos a versão otimizada da heurística VL, implementada utilizando-se a linguagem de programação C (conforme já citamos previamente). Todos os testes apresentados neste relatório foram repetidos dez vezes cada (sendo apresentadas as médias dos mesmos), em máquinas com processador *Pentium IV Dual Core 3.0 GHz - 64 bit* e 1 GB de memória RAM, utilizando *Linux 2.6*.

Temos duas observações importantes sobre os dados apresentados:

- as análises referentes ao tempo de execução são medidas reais de tempo (chamado "tempo de relógio").

- os valores apresentados como total de padrões encontrados referem-se, em todos os casos, ao total de padrões maximais.

Como definição, padrões maximais são aqueles que não podem ser classificados como um subconjunto de algum outro padrão encontrado. A escolha por padrões maximais possui como base uma tentativa de avaliar os padrões mais significativos da base, além de diminuir muito o volume de informações processado (basta lembrar que, para cada

padrão freqüente f maximal de n trincas, existiram outros $2^n - 2$ padrões freqüentes não-maximais que são subconjuntos de f).

5.2.2 Impacto dos parâmetros na heurística

A heurística desenvolvida possui uma série de parâmetros que controlam o comportamento de sua execução. Ao longo desta subseção, iremos estudar, principalmente, a variação no tempo e no total de padrões encontrados de acordo com um desses parâmetros, fixando os demais.

Conforme dissemos, a conseqüência da definição da constante MAX, para limitar o número de mapeamentos de cada cadeia na árvore de trincas, foi atribuir um comportamento não-determinista à nossa heurística. Dessa forma, para realizar um estudo sobre como a enumeração de mapeamentos feita afeta os resultados, apresentamos uma tabela com os desvios-padrão para os dados de todos os gráficos presentes nos experimentos para avaliar cada parâmetro da heurística.

5.2.2.1 Suporte mínimo

Primeiramente, analisaremos o comportamento da heurística com a variação do suporte mínimo utilizado. Neste experimento, variamos o suporte mínimo, para considerarmos um padrão como freqüente, entre valores de 140 a 190. Para os demais parâmetros, utilizamos: uma profundidade limite igual a 9, a constante MAX com o valor 200 e a constante $FATOR$ igual a 2. A razão de usarmos tais valores será esclarecida ao longo dos demais experimentos.

O gráfico da Figura 5.2 apresenta o impacto no tempo de execução, enquanto o gráfico da Figura 5.3 mostra a quantidade de padrões (cadeias de trincas) freqüentes maximais encontrados. Podemos perceber que o aumento do suporte restringe muito o número de padrões freqüentes distintos encontrados, diminuindo, por exemplo, de 25.312 para 1.116 nos experimentos para a Base Completa. Essa restrição, obviamente, diminui o tempo de execução, pois apenas um número pequeno de cadeias de tamanho igual a um (ou seja, que possuem apenas uma trinca) existirão na árvore.

Outra característica importante desses gráficos refere-se ao comportamento dos experimentos com a Base Resumida. Apesar de possuir um total de arestas quase dez vezes menor que o da Base Completa, os tempos de execução e os números de padrões maximais encontrados foram próximos para ambas. Atribuímos essa proximidade no total de padrões, principalmente para suportes mínimos maiores, à presença dos principais fóruns nas duas bases, que são aqueles capazes de gerar cadeias suficientemente freqüentes. Já para a Base Pós, era esperado que tanto o número de padrões quanto

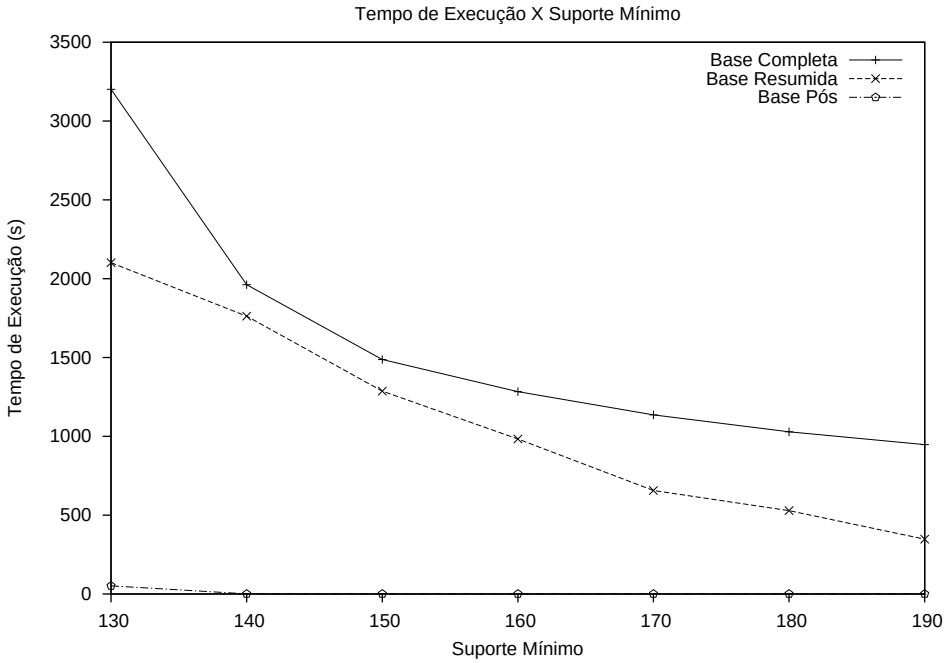

Figura 5.2. Variação do tempo em função do suporte mínimo.

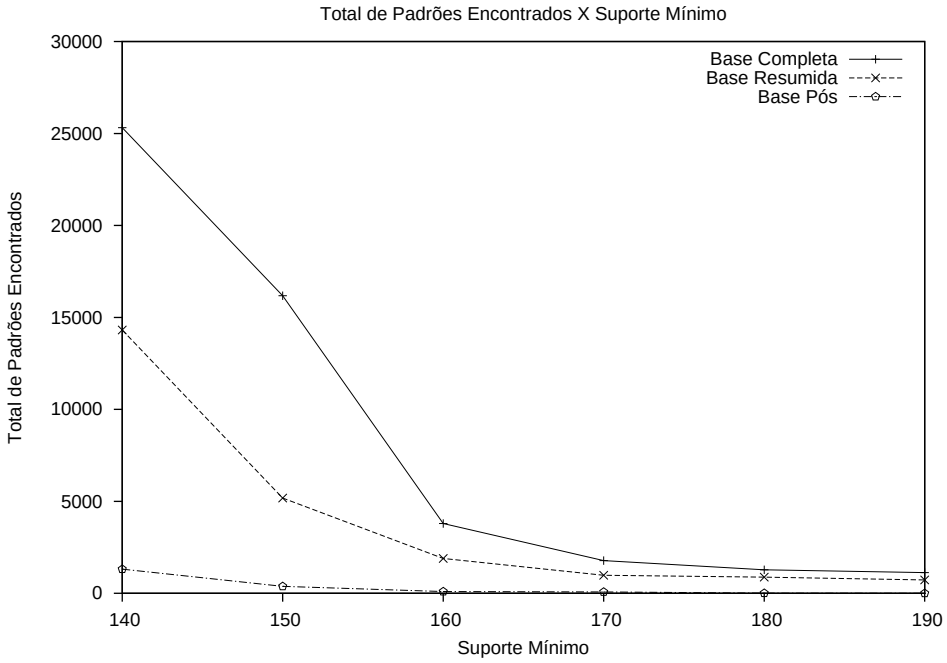

Figura 5.3. Quantidade de padrões maximais em função do suporte mínimo.

	Base Completa		Base Resumida		Base Pós	
Suporte	Tempo	Total	Tempo	Total	Tempo	Total
130	679,08	12110,38	576,63	8130,98	1,77	459,30
140	568,67	7543,34	379,62	5493,54	0,07	124,55
150	279,59	5423,91	91,86	1032,91	0,10	72,88
160	116,13	578,32	25,29	320,49	0,10	17,21
170	97,74	345,21	19,66	132,33	0,04	13,09
180	50,11	190,83	19,55	121,11	0,05	1,02
190	22,4	184,78	16,96	102,29	0,09	0,33

Tabela 5.5. Desvios-padrão para a variação do suporte mínimo.

o tempo de execução fossem consideravelmente menores, ao serem comparados com os resultados das outras bases.

Para os demais testes, escolhemos o valor 150 para fixar o suporte mínimo. Achamos esse valor interessante, pois permite a geração de um grande número de padrões, enquanto o tempo de execução ainda é abaixo de meia hora.

Na Tabela 5.5, apresentamos os desvios-padrão de todos os pontos dos gráficos da Figura 5.2 e da Figura 5.3. Podemos notar que os valores dessa tabela são consideravelmente altos. Destacamos que, para a Base Completa, devido ao seu grande número de nodos e arestas, o impacto da enumeração não determinista de mapeamentos parece ser agravado. Por fim, lembramos que os valores de desvio-padrão pequenos para a Base Pós não foram obtidos devido a uma possível eficiência da heurística, mas sim devido ao fato de que os valores de tempo e execução são ínfimos, já que o suporte utilizado é inadequado para tais dados.

5.2.2.2 Profundidade máxima

Analisaremos, agora, como o comportamento da heurística é afetado ao utilizarmos diferentes valores para a profundidade máxima da busca em profundidade. Lembramos que esse parâmetro é, juntamente com a impossibilidade de gerar novas cadeias válidas, a condição de parada da recursividade do procedimento de busca, conforme mostrado no Capítulo 4. Assim, as figuras 5.4 e 5.5 apresentam o tempo de execução e o total de padrões, respectivamente, variando-se os valores da profundidade máxima de 2 a 11. Os demais parâmetros foram fixados em: suporte mínimo igual a 150, constante MAX igual a 200 e constante $FATOR$ igual a 2.

Em relação ao tempo de execução, podemos perceber que, tanto para a Base Completa quanto para a Base Resumida, existe uma tendência de crescimento à medida que aumentamos os valores da profundidade, como esperado. Além disso, os valores de ambas as bases são muito próximos. Por outro lado, quando contabilizamos o total

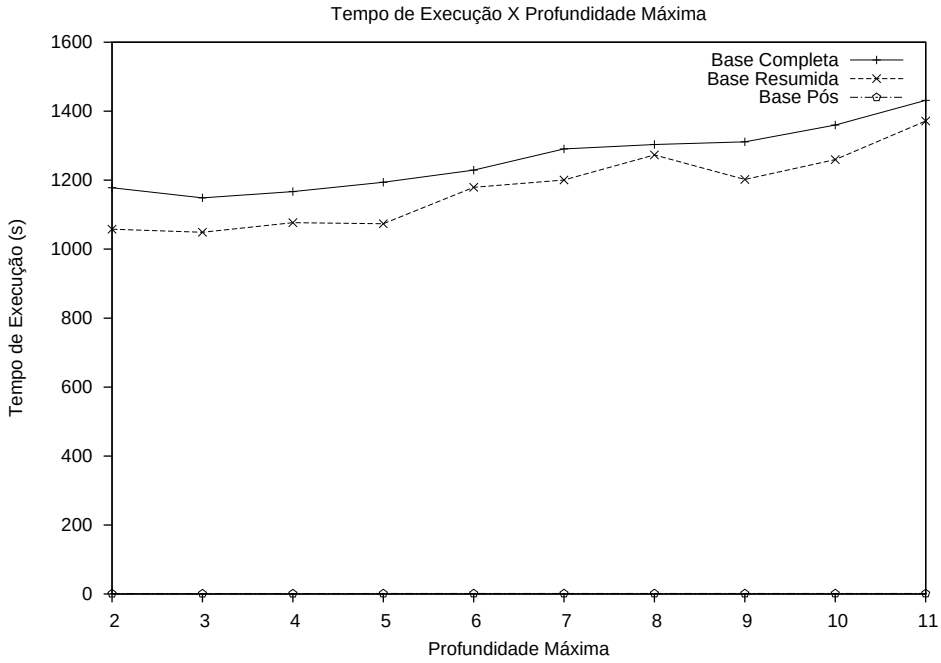

Figura 5.4. Comportamento do tempo com a variação da profundidade máxima.

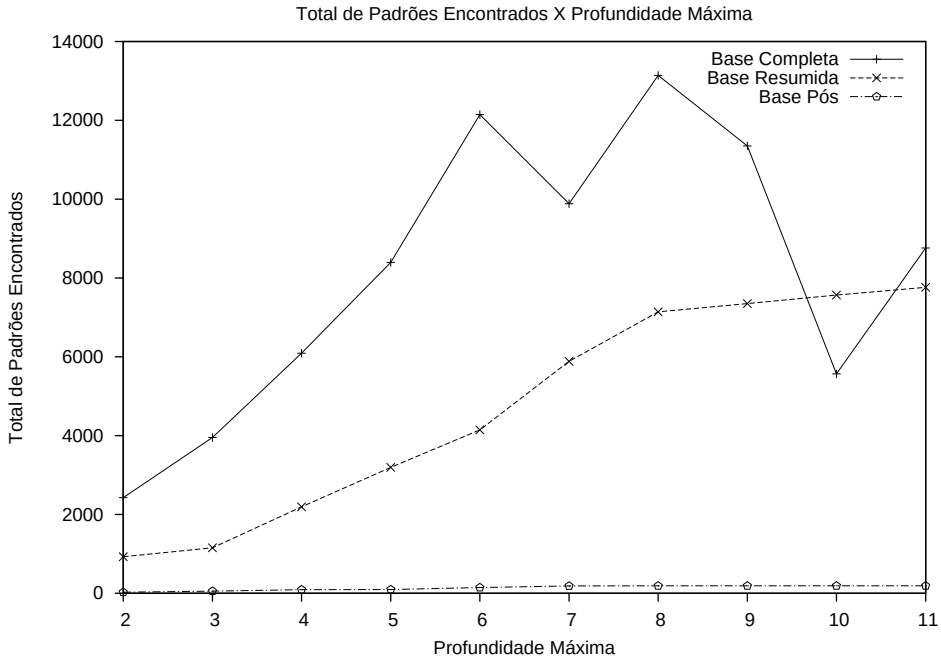

Figura 5.5. Quantidade de padrões freqüentes pela da profundidade máxima.

	Base Completa		Base Resumida		Base Pós	
Profundidade	Tempo	Total	Tempo	Total	Tempo	Total
2	207,32	374,31	128,84	63,97	0,02	5,31
3	103,97	489,99	102,97	70,18	0,03	13,78
4	128,84	743,56	109,77	234,77	0,04	17,64
5	132,19	945,66	112,11	384,32	0,05	16,32
6	173,44	1221,43	153,63	399,88	0,04	19,21
7	183,9	1023,38	162,28	596,54	0,03	21,33
8	232,74	1439,21	173,42	765,67	0,07	22,98
9	267,21	1099,32	162,98	754,96	0,04	22,43
10	254,32	1593,12	164,83	775,12	0,05	24,91
11	273,12	8790,23	171,96	828,21	0,04	21,74

Tabela 5.6. Desvios-padrão para a variação da profundidade máxima.

de padrões maximais, nota-se que o gráfico da Base Completa apresenta pontos em que a curva possui um declínio acentuado, sendo os ponto de profundidade 7 e 10 os que chamam mais atenção. Analisando os padrões encontrados nesse experimento, foi possível perceber que essa queda deve-se, principalmente, a quais padrões foram gerados. Por serem maximais, os padrões de tamanho igual a 10, nessa base, eliminaram muitos padrões de tamanho igual a 9, encontrados no experimento utilizando essa profundidade limite. Essa justificativa ajuda, inclusive, a entender o porquê de o tempo de execução apresentar um crescimento ao aumentarmos a profundidade máxima, já que apesar de encontrarmos menos padrões maximais, precisamos também processar os padrões que não serão considerados maximais. A queda para o ponto de profundidade máxima 7 segue a mesma linha de raciocínio, com a ressalva de que, no caso, ainda existiam muitas possibilidades para a geração de novos padrões, evitando assim uma diferença maior no total de padrões maximais. O comportamento anômalo da Base Completa não foi observado nas bases Resumida e Pós, onde houve um crescimento (em taxas decrescentes) do número total de padrões encontrados a cada aumento do limite de profundidade da busca.

Portanto, para os demais testes, escolheremos o valor 9 como padrão para a profundidade máxima. Tal escolha é justificada por dois motivos: o primeiro é o fato de, assim, evitarmos um ponto de queda acentuada no total de padrões (tamanho 10) na Base Completa; o segundo refere-se aos experimentos com o algoritmo *Apriori*, mostrados adiante, cujos padrões obtidos de maior tamanho possuem 9 elementos.

Na Tabela 5.6, apresentamos os desvios-padrão de todos os pontos dos gráficos nas figuras 5.4 e 5.5. Se comparados aos desvios da Tabela 5.5, notamos que as dimensões são ligeiramente menores, apesar de esses desvios ainda possuírem valores altos. O ponto de profundidade máxima igual a 10 para a Base Completa, citado previamente,

possui um desvio muito próximo ao ponto 9, corroborando com nossas análises.

5.2.2.3 Constante MAX

O próximo impacto a ser estudado refere-se à constante MAX definida no Capítulo 4, que indica o máximo de mapeamentos que armazenamos para cada cadeia representada na árvore de trincas. A Figura 5.6 apresenta como essa constante afeta o tempo de execução, enquanto a Figura 5.7 revela a variação no total de padrões maximais encontrados. Os valores de MAX variaram entre 160 e 230, sendo os demais parâmetros fixados como: suporte mínimo igual a 150, profundidade máxima igual a 9 e a constante $FATOR$ com valor 2.

Em ambos os gráficos, podemos facilmente notar que o número máximo de mapeamentos permitido restringe os resultados da heurística, já que valores maiores desse parâmetro permitem que a heurística encontre números crescentes de padrões freqüentes (com o tempo aumentando na mesma proporção). Outro fator interessante é que tal característica foi comprovada em todas as três bases de dados, demonstrado que o efeito ocasionado por MAX (e a correspondente enumeração não determinista de mapeamentos) aparece de forma considerável e independente de demais fatores.

Para os testes dos outros parâmetros, escolhemos 200 como valor a ser fixado para a constante MAX. Dessa forma, temos um valor que é capaz de gerar um grande

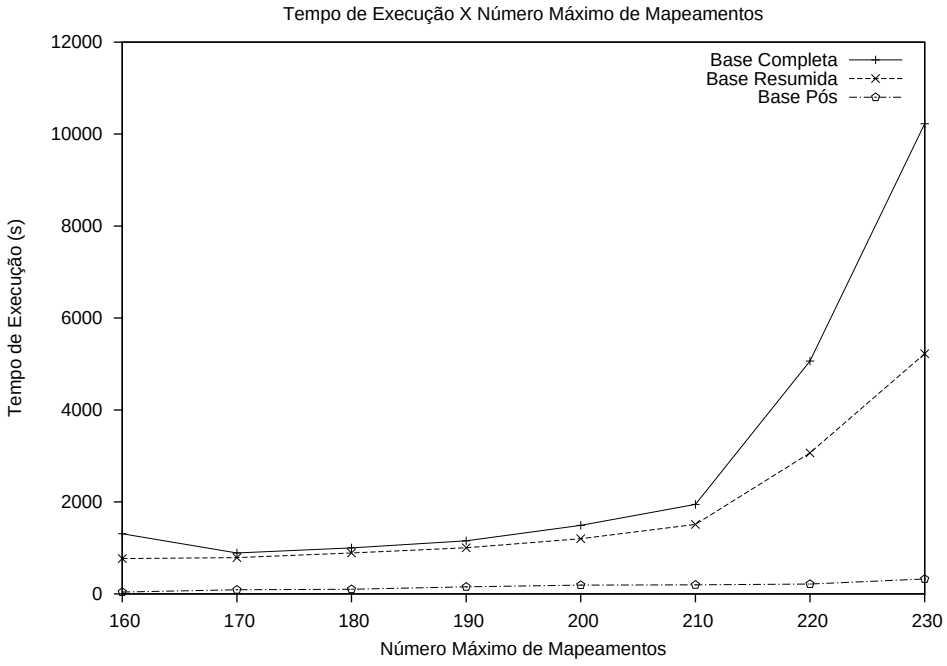

Figura 5.6. Comportamento do tempo com a variação do máximo de mapeamentos.

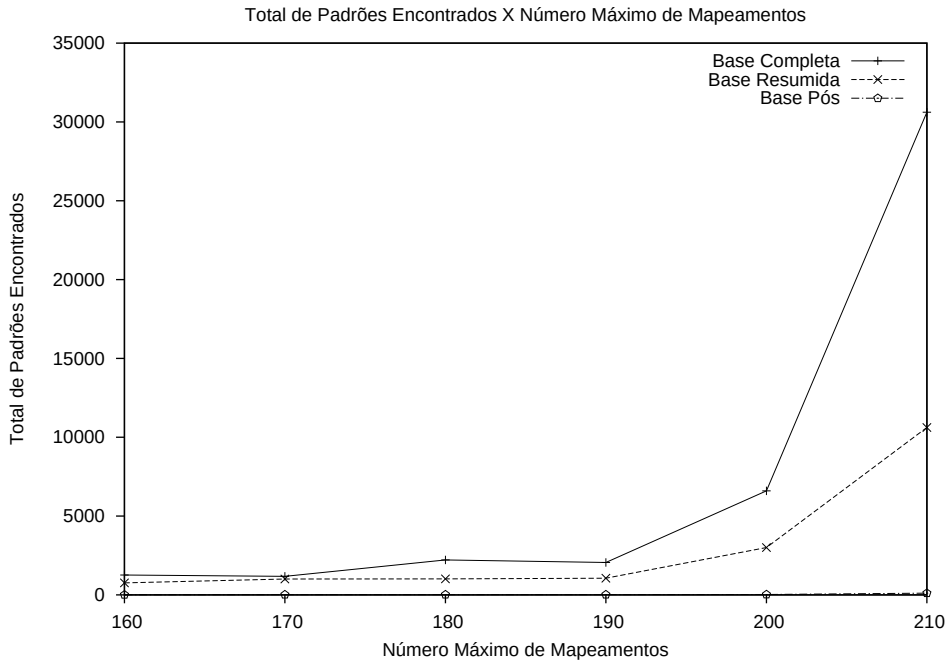

Figura 5.7. Variação do total de padrões em função do máximo de mapeamentos.

	Base Completa		Base Resumida		Base Pós	
MAX	Tempo	Total	Tempo	Total	Tempo	Total
160	290,89	104,34	39,32	44,84	2,95	1,23
170	148,29	138,78	42,18	87,50	5,43	1,19
180	129,32	234,41	88,87	92,35	9,84	1,34
190	131,90	224,35	93,43	102,32	9,90	1,98
200	145,87	763,23	109,57	284,95	15,76	2,49
210	293,71	4854,62	163,98	1087,67	17,93	9,85
220	602,34	15165,13	308,17	2543,21	20,12	11,21
230	1223,48	41221,68	536,78	7328,99	38,23	19,68

Tabela 5.7. Desvios-padrão para a variação da constante MAX.

número de mapeamentos, mas que o tempo de execução não é inviável.

Na Tabela 5.7, apresentamos os desvios-padrão de todos os pontos dos gráficos nas figuras 5.6 e 5.7. Destacamos que, quanto maior é o número de padrões sendo encontrados, maior torna-se o desvio. Isso mostra que o impacto da enumeração não determinista, ocasionado justamente pela constante MAX, é relevante e não pode ser desconsiderado.

5.2.2.4 Constante $FATOR$

O último dos parâmetros da heurística a ser estudado é relativo à constante $FATOR$, também definida no Capítulo 4. Essa constante define o impacto do corte feito durante o processo de expansão, que é o responsável por gerar as novas cadeias encontradas na busca em profundidade pela árvore de trincas. Para realizarmos essa análise, utilizamos para a constante valores de 1 a 10 (vale lembrar que, por definição, essa constante deve possuir valores pequenos, para manter linear a complexidade da expansão). O suporte mínimo foi fixado em 150, a profundidade limite em 9 e a constante MAX em 200.

As figuras 5.8 e 5.9 apresentam, então, a variação do tempo e do total de padrões maximais minerados, respectivamente, em função do valor da constante MAX. Percebe-se que ambos os gráficos apresentam um comportamento crescente. A principal característica a ser ressaltada, para as três bases, é a grande diferença ao utilizarmos o valor 1 ou o valor 2 para a constante. O valor 1 tende a gerar resultados muito ruins (poucos padrões), pois, conforme discussão do capítulo anterior, limita muito as iterações do processo de expansão e geração de novas cadeias. Para o valor 2, permite-se gerar até o dobro de MAX elementos no processo citado, para então verificarmos a validade dos mapeamentos.

Ao contrário dos demais parâmetros, onde os resultados da Base Resumida foram muito próximos aos da Base Completa, nesse experimento podemos observar que o

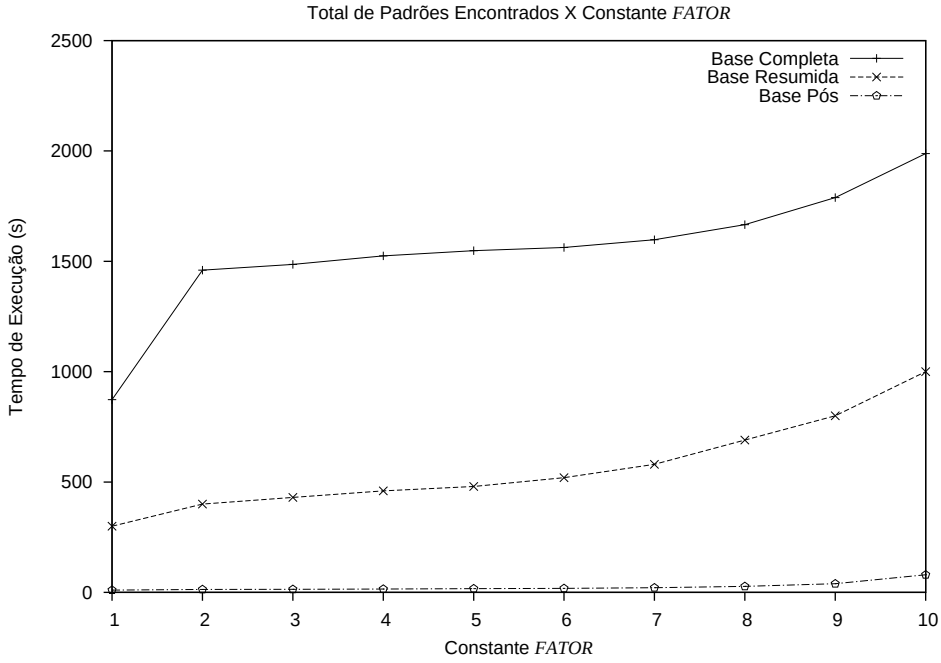

Figura 5.8. Comportamento do tempo com a variação de $FATOR$.

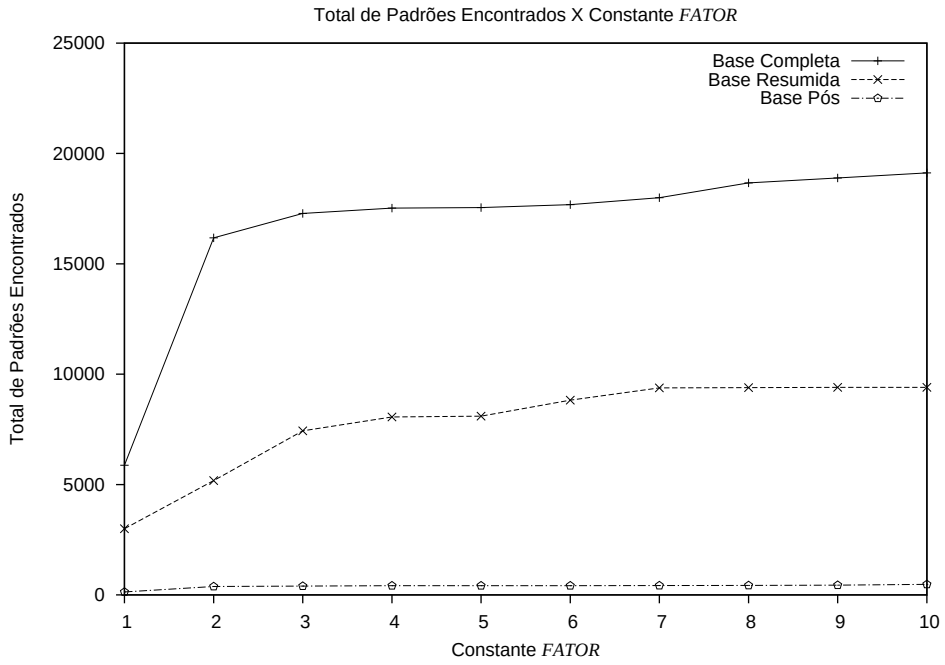

Figura 5.9. Variação do total de padrões em função de $FATOR$.

comportamento foi semelhante, mas em termos da magnitude dos valores, as bases apresentam clara distinção. Acreditamos que esse fato ficou mais destacado nesse experimento, pois a constante $FATOR$ tem seu efeito diretamente sobre o processo de geração de mapeamentos de uma nova cadeia. Se aumentarmos a capacidade (através do aumento do número de iterações dos laços respectivos) desse processo, permitimos que mais mapeamentos distintos sejam criados. Isso tende a beneficiar a base com mais arestas, que possui maior probabilidade de gerar mapeamentos válidos.

O valor para ser utilizado como padrão para a constante $FATOR$, em outros experimentos, foi 2. Podemos justificar esse valor pelo fato de que a idéia da criação dessa constante era utilizar o menor valor possível e, além disso, pode-se perceber que, segundo o gráfico, valores maiores que 2 alteram pouco os resultados.

Na Tabela 5.8, apresentamos os desvios-padrão de todos os pontos dos gráficos nas figuras 5.8 e 5.9. Tanto para o tempo de execução, quanto para o total de padrões encontrados, os desvios seguem comportamentos muito semelhantes, seguindo o mesmo ritmo crescente apresentado nos gráficos.

5.2.2.5 Sobre a enumeração não determinista de mapeamentos

Para cada um dos quatro parâmetros de execução da heurística, estudados acima, apresentamos uma tabela com os seus respectivos desvios-padrão. Em todos, podemos perceber que os valores foram muito elevados. Isso significa que execuções distintas

	Base Completa		Base Resumida		Base Pós	
$FATOR$	Tempo	Total	Tempo	Total	Tempo	Total
1	20,31	199,87	22,98	270,87	0,03	5,43
2	128,29	1563,84	37,21	411,21	0,01	13,21
3	126,93	1488,32	46,87	760,93	0,01	16,59
4	154,57	1723,70	40,12	750,29	0,05	17,44
5	167,06	1827,19	49,32	823,64	0,03	15,98
6	169,99	1902,21	61,21	910,08	0,04	16,09
7	173,76	1989,03	59,77	899,54	0,11	23,32
8	172,21	1954,94	78,90	9409,76	0,09	23,09
9	174,77	2001,90	86,30	9594,99	0,27	25,21
10	203,89	2238,01	131,46	9904,23	1,01	31,98

Tabela 5.8. Desvios-padrão para a variação da constante $FATOR$.

produzem resultados muito distintos, em termos de tempo de execução e total de padrões maximais encontrados. Tal fato ocorre pelo efeito não determinista durante a enumeração dos mapeamentos, causado pela inserção da constante MAX na heurística, conforme discussão do Capítulo 4.

Esse resultado já era esperando, pois, através de cortes de mapeamentos escolhidos de forma aleatória, podemos eliminar possibilidades vitais para a continuidade da busca em profundidade. Destacamos, então, que uma das principais melhorias a serem feitas na heurística refere-se aos procedimentos de enumeração não determinista de mapeamentos. Por isso, a citamos como trabalho futuro no próximo capítulo.

5.2.3 Impacto do suporte para o *Apriori*

Nesta subseção, analisaremos rapidamente o comportamento do algoritmo *Apriori*, ao ser utilizado para encontrar padrões nas três bases citadas neste capítulo. Primeiramente, iremos citar, de forma sucinta, algumas características desse algoritmo para, em seguida, ressaltar a diferença entre os conceitos de suporte mínimo do mesmo e o de nossa heurística. Por fim, apresentamos dois gráficos relativos ao comportamento do *Apriori*, com a variação do suporte.

O *Apriori* é um algoritmo clássico de geração e aprendizagem de regras de associação, que opera sobre um conjunto de transações, ou conjunto de itens. O algoritmo tenta encontrar subconjuntos de itens que são comuns em, pelo menos, δ transações distintas. Esse número é exatamente o suporte mínimo. Ele utiliza uma abordagem *bottom up*, onde subconjuntos freqüentes são estendidos em um item por vez, assim como nossa heurística. Entretanto, ao contrário do nosso trabalho, o *Apriori* usa uma busca em largura durante seu processo de geração de candidatos. Embora importante,

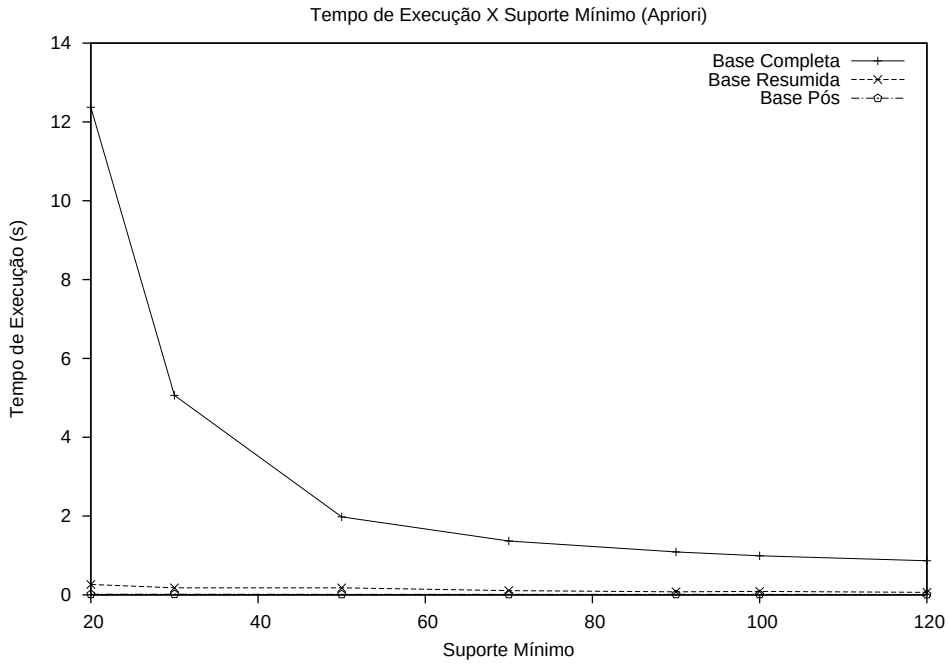

Figura 5.10. Variação do tempo em função do suporte mínimo no *Apriori*.

esse algoritmo sofre de um número significativo de ineficiências, como a grande quantidade de candidatos gerados e o elevado número de passos até gerar padrões com muitos elementos.

Podemos perceber, então, que o conceito de suporte da nossa heurística e do *Apriori* são muito distintos. Embora, para ambos, o suporte mínimo indique um limiar para distinguirmos se um padrão é ou não freqüente, para a construção de agrupamentos de conferências, o significado de um valor x de suporte significa:

- para o *Apriori*, que x ou mais autores distintos devem ter publicado pelo menos um artigo em todas as conferências presentes no padrão, para que este último seja freqüente.

- para a heurística, que existe x ou mais conjuntos de autores (de tamanho variável) que estejam interligados por arestas que representem as conferências do padrão (de maneira tal que um subgrafo conexo seja formado).

A partir dessa definição, notamos como é difícil comparar os resultados de ambas as técnicas. Por isso, é feita a análise desta subseção. Outro detalhe importante a ser citado é que, assim como nos outros experimentos, analisaremos apenas os padrões freqüentes que forem maximais.

Apresentaremos, agora, a Figura 5.10 e a Figura 5.11. Nelas, estão os gráficos com a variação do tempo e do total de padrões freqüentes maximais encontrados,

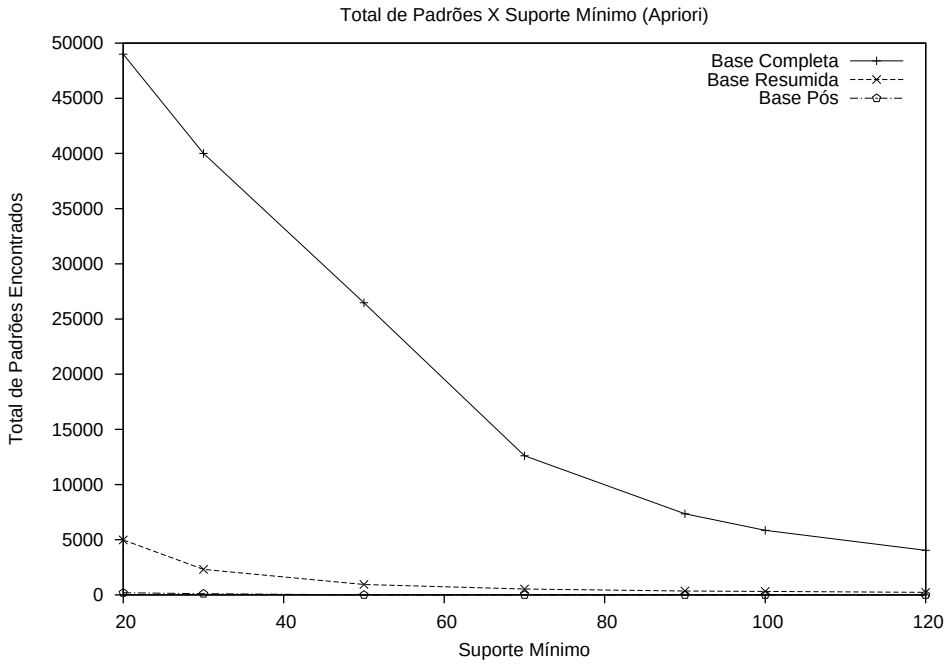

Figura 5.11. Quantidade de padrões maximais em função do suporte mínimo no *Apriori*.

respectivamente, de acordo com o suporte mínimo utilizado para o algoritmo *Apriori*. É possível notar que, quanto menores os valores de suporte, maiores serão os padrões encontrados, o que eleva muito o tempo de execução. Isso se deve ao fato de que cada iteração da busca em largura é muito dispendiosa, custo esse que aumenta a cada nível de profundidade.

Analisando-se esses gráficos, optamos por utilizar o valor de suporte do *Apriori* igual a 30. Esse valor será utilizado para realizar a análise da qualidade dos agrupamentos, ao compararmos os resultados do algoritmo *Apriori* com os resultados dos experimentos realizados, em que todos os parâmetros da heurística possuem valores fixos, escolhidos e justificados nas subseções anteriores. Essa escolha é devida à proximidade do número de padrões encontrados para ambos os algoritmos.

Ressaltamos novamente que não estabelecemos uma comparação direta entre os gráficos das figuras 5.2 e 5.3 e os das figuras 5.10 e 5.11, pois os conceitos de suporte são distintos em cada caso. Com o intuito de realizar tal comparação, a subseção a seguir irá definir e avaliar novas métricas.

5.2.4 Comparação da heurística com o *Apriori*

Nesta subseção, iremos apresentar uma série de comparações entre a heurística desenvolvida e o *Apriori*. Essas comparações possuem como principal objetivo avaliar a

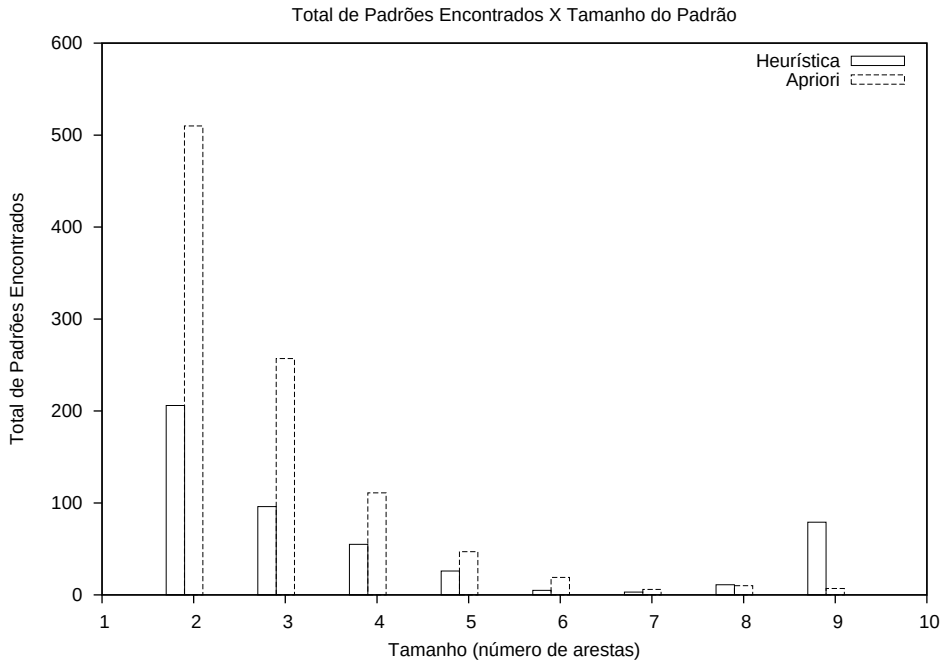

Figura 5.12. Comparação da distribuição de padrões entre a heurística e o *Apriori*.

qualidade dos resultados obtidos. Para realizar essa tarefa, apresentaremos agora uma série de experimentos, realizados exclusivamente sobre os dados da Base Resumida. Conforme citamos no início deste capítulo, essa base, ao contrário das demais, possui a informação da área associada a cada conferência. Essa restrição torna-se óbvia para os experimentos relativos à qualidade dos padrões, apresentados mais adiante.

A primeira comparação a ser realizada é em relação à distribuição do número de padrões encontrados para cada tamanho dos mesmos. Na Figura 5.12, temos o gráfico com os dados para o *Apriori*, provenientes de execuções com suporte igual a 30, e os dados da heurística, provenientes de execuções com suporte mínimo 150, profundidade limite 9, constante MAX 200 e constante $FATOR$ 2. É interessante notar que o comportamento apresentado por ambos, onde o número de padrões tem uma tendência de decrescimento, é justificável, pois estamos analisando os padrões maximais. Entretanto, podemos observar que a nossa heurística encontra um número grande de padrões de tamanho 9, contrariando essa tendência. Isso acontece, porque, ao contrário do *Apriori*, nossa heurística conseguiria encontrar padrões com mais de nove elementos, para os mesmos valores dos parâmetros.

Até o momento, os experimentos analisaram apenas métricas globais (total de padrões e tempos), sendo que nenhum analisou o conteúdo dos padrões sendo retornados pela heurística. Como a Base Resumida possui a informação da área das conferên-

cias, podemos analisar cada um dos padrões de um resultado e, assim, comparar nossa heurística e o *Apriori*, em termos de qualidade. Nos gráficos a serem apresentados, analisamos o comportamento de ambos os algoritmos conforme a variação do suporte mínimo utilizado. Os demais parâmetros para a heurística foram determinados como 9 para a profundidade limite, 200 para a constante MAX e 2 para a constante $FATOR$.

Para esses experimentos, então, definimos uma métrica de qualidade da seguinte forma:

> Quanto maior o número de padrões maximais, com mais de um elemento, que possua todas as suas conferências associadas a uma mesma área, melhor é o resultado do algoritmo.

A partir de agora, esses padrões que participam da métrica de qualidade serão denominados padrões-área. Além dessa métrica, analisaremos, também, qual a porcentagem de padrões-área nos resultados e quantos fóruns distintos são associados a padrões-área, ou seja, foram corretamente classificados para a área respectiva. Ambas as métricas (total de padrões-área e sua porcentagem) foram definidas de maneira que nos permitisse aplicar os conceitos tradicionais de precisão e revocação, respectivamente, em nosso trabalho.

Para exemplificar, vamos supor que uma execução gerou como resultado quatro padrões freqüentes maximais distintos, citados a seguir:

1. Conferência A, Conferência B, Conferência C.
2. Conferência B, Conferência D.
3. Conferência C, Conferência D, Conferência E.
4. Conferência D, Conferência F, Conferência G.

Se as conferências A, B, C e D pertencerem a uma área e as conferências E, F e G a outra, teremos um total de dois padrões-área no resultado (os dois primeiros), o que equivale a uma porcentagem de $0,50$. Além disso, o total de fóruns distintos identificados é igual a 4 (A, B, C e D).

Nas figuras 5.13 e 5.14, temos, respectivamente, um gráfico com resultados obtidos por nossa heurística e outro com os do *Apriori*. Para ambos, analisamos o total de padrões-área encontrados e a porcentagem deles em relação à quantidade total de padrões maximais obtidos. Sobre o total de padrões, percebemos que a heurística encontra um maior número de áreas e, além disso, apresenta um decrescimento menos acentuado, embora parecido com o do algoritmo *Apriori*. Em relação à porcentagem, enquanto para o *Apriori*, os valores da porcentagem parecem oscilar entre 18% e 22%

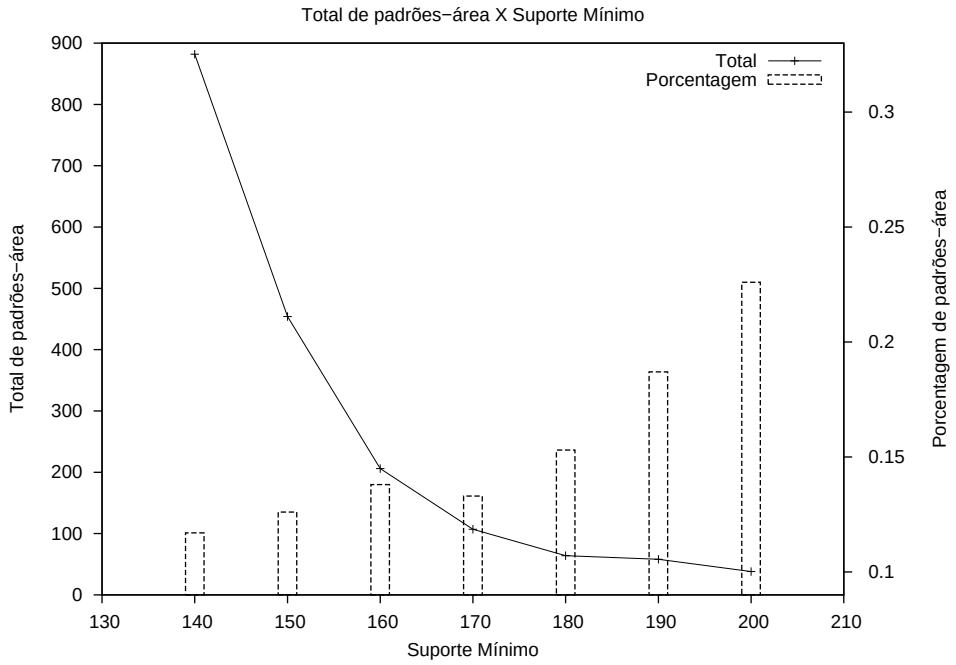

Figura 5.13. Total de padrões-área encontrados pela heurística.

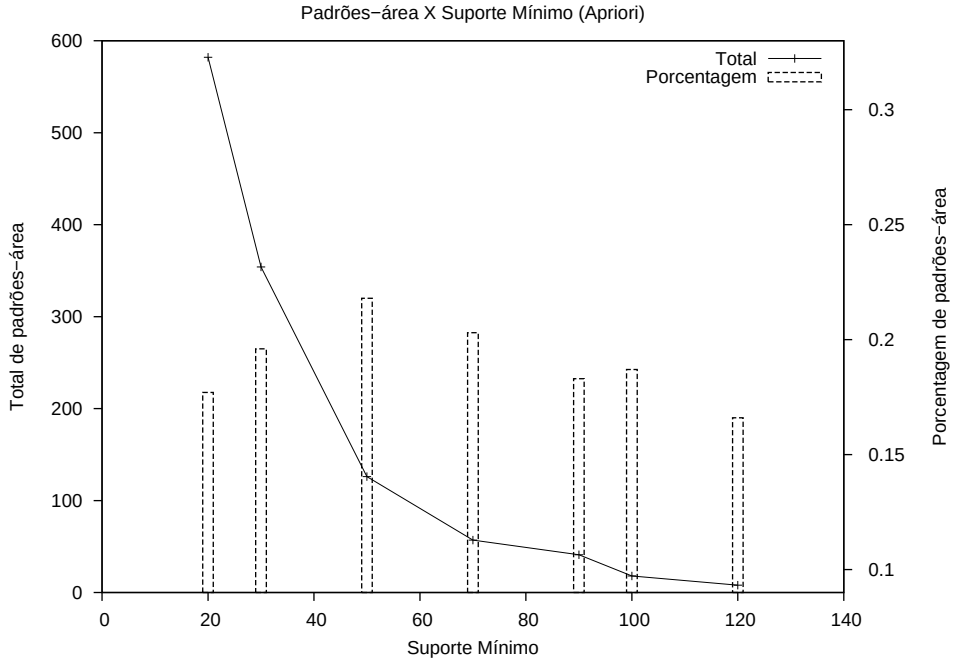

Figura 5.14. Total de padrões-área encontrados pelo *Apriori*.

(sem padrão aparente), para a heurística, percebe-se claramente uma tendência de aumento da porcentagem com o aumento do suporte. Podemos explicar esse fato devido à capacidade da nossa estratégia de analisar relacionamentos indiretos e, através da

filtragem de padrões inconsistentes ocasionada por valores altos de suporte, aumentar a porcentagem de padrões-área.

Após analisarmos a quantidade de padrões-área e sua porcentagem em relação ao resultado total de ambos os algoritmos, vamos, para o mesmo experimento, analisar quais conferências estão presentes nesses padrões. Assim, realizamos o seguinte procedimento: criamos um conjunto com a união dos fóruns presentes em cada um dos padrões-área e, dessa forma, a cardinalidade desse conjunto identifica quantas conferências foram corretamente classificadas.

O gráfico da Figura 5.15 e o da Figura 5.16 contêm exatamente tal informação, para a heurística e para o *Apriori*, respectivamente. Assim, pode-se perceber que, para o total de padrões-área identificados, o número de fóruns distintos classificados corretamente foi superior nos resultados obtidos pela heurística (para os maiores suportes avaliados, a heurística VL identificou 39 conferências distintas, enquanto o *Apriori* apenas 9).

Embora não seja possível garantir, através desses gráficos, que a heurística produzirá sempre melhores resultados no processo de classificação do que o *Apriori*, demonstramos que a estratégia de modelar os dados como multigrafos permitiu gerar resultados tão bons ou melhores que um dos algoritmos clássicos utilizados para o problema de classificação e agrupamento. Além disso, com algumas otimizações, os resultados poderiam ser ainda melhores e, por isso, a confirmação do potencial da utilização da

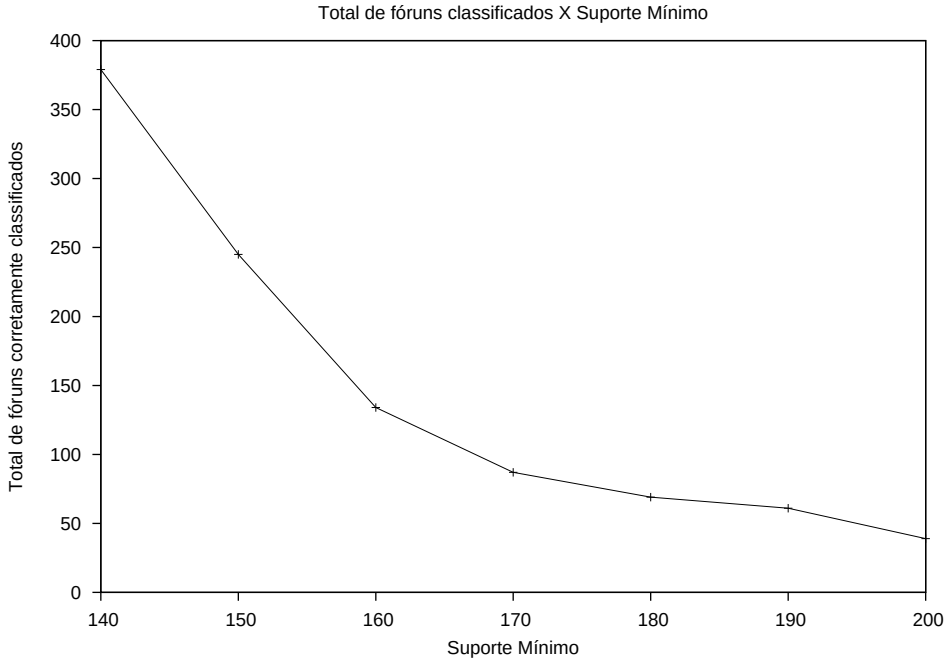

Figura 5.15. Total de fóruns em padrões-área encontrados pela heurística.

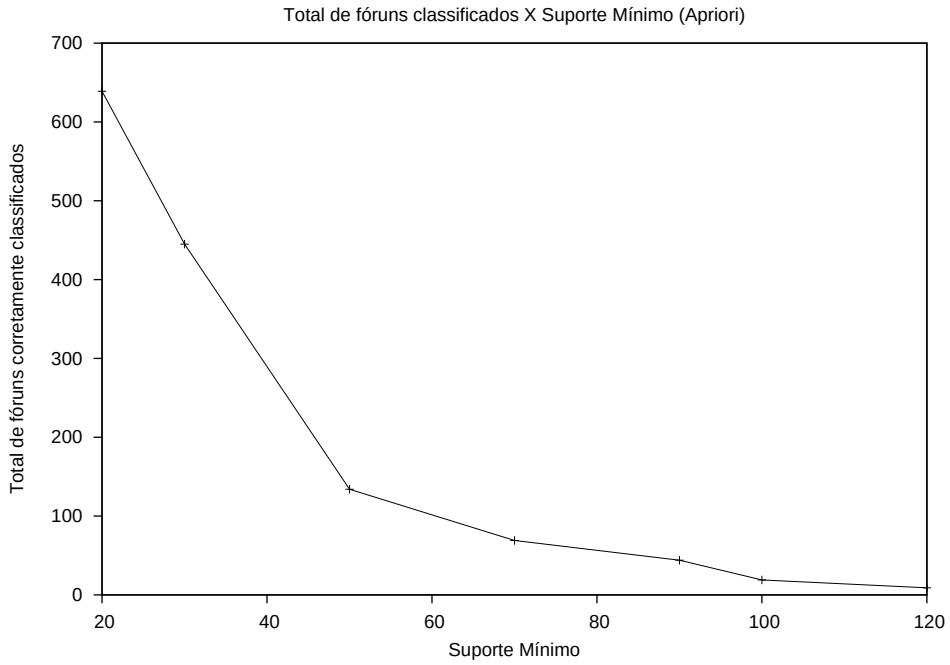

Figura 5.16. Total de fóruns em padrões-área encontrados pelo *Apriori*.

heurística VL em trabalhos futuros.

Nossos experimentos terminam aqui. Através deles, foi possível analisar vários aspectos da heurística de Vizinhança Limitada. No capítulo a seguir, destacaremos alguns pontos principais obtidos como conclusões desta dissertação, além de possibilidades de produzir novos trabalhos com base no que foi apresentado.

Capítulo 6

Conclusões e Trabalhos Futuros

Nesta dissertação, apresentamos a heurística VL, capaz de minerar bases de dados relacionais. Utilizamos como aplicação o problema de classificação conferências, a partir de redes de co-autoria. Nossa heurística foi baseada em técnicas inovadoras, utilizando conceitos de mineração de multigrafos para explorar relacionamentos múltiplos e indiretos presentes nos dados.

Para desenvolvermos uma estratégia capaz de minerar padrões freqüentes em bases de dados relacionais, definimos uma série de conceitos e fundamentos que nos permitiram, a partir da modelagem dos dados em multigrafos, extrair informações essenciais, ignoradas por outras técnicas existentes. Dessa forma, obtivemos uma base teórica sobre a qual diversas aplicações reais podem ser modeladas, de forma a preservar relações importantes existentes nos dados.

Além de apresentarmos a heurística, demonstramos também sua utilização em um cenário real: redes de co-autoria. Através de um conjunto variado de experimentos sobre grandes massas de dados, obtivemos resultados que permitiram verificar que as áreas de conhecimento encontradas são relevantes, sendo equiparáveis aos resultados do algoritmo *Apriori*. Dessa forma, comprovamos a capacidade de uso e eficácia da técnica proposta na modelagem de problemas desse tipo.

Existem várias possibilidades de continuidade para o trabalho apresentado nessa dissertação. Dentre elas, destacamos:

- Aplicação da heurística VL para, além de definir áreas, classificar autores que participam das mesmas, tentando melhorar a capacidade de identificar fóruns nas redes de co-autoria.

- Utilização da estratégia apresentada para encontrar padrões freqüentes em outros cenários, diferentes das redes de co-autoria. Citamos dois exemplos: o problema de ambigüidade de nomes, onde indivíduos poderiam ser discriminados a partir

de padrões de relacionamentos; e o problema de identificação de doenças, onde a partir de uma rede formada por sintomas apresentados por um conjunto de pacientes poderiam ser utilizados para identificar doenças, o que caracterizaria uma forma inovadora de diagnóstico.

- Estudo de alternativas mais elaboradas para o processo de enumeração não determinista de mapeamentos das cadeias, de forma a reduzir o impacto desse procedimento nos resultados obtidos.

- Realização de um processo de "estratificação", aplicando a heurística sobre resultados gerados por ela mesmo e previamente tratados (por exemplo, substituindo padrões freqüentes por um nodo simbólico).

- Comparação dos resultados da heurística VL com outras técnicas existentes na literatura, diferentes do *Apriori*, permitindo uma compreensão maior sobre o potencial obtido por explorarmos relacionamentos múltiplos e indiretos.

Referências Bibliográficas

[1] An, Y.; Janssen, J. e Milios, E. E. (2004). Characterizing and mining the citation graph of the computer science literature. *Knowl. Inf. Syst.*, 6(6):664–678.

[2] Castro, D. e Grossman (1999). Famous trails to paul erdos. *MATHINT: The Mathematical Intelligencer*, 21.

[3] Chen, C. e Carr, L. (1999). Trailblazing the literature of hypertext: author co-citation analysis. In *HYPERTEXT '99: Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots*, pp. 51–60, New York, NY, USA. ACM.

[4] Cook, D. J. e Holder, L. B. (2006). *Mining Graph Data*. John Wiley & Sons.

[5] Cunningham, S. e Dillon, S. (1997). Authorship patterns in information systems. *Scientometrics*, 39(1):19–27.

[6] Dehaspe, L. e Toivonen, H. (1999). Discovery of frequent datalog patterns. *Data Min. Knowl. Discov.*, 3(1):7–36.

[7] Farkas, I.; Derenyi, I.; Jeong, H.; Neda, Z.; Oltvai, Z. N.; Ravasz, E.; Schubert, A.; Barabasi e Vicsek, T. (2002). Networks in life: scaling properties and eigenvalue spectra. *Physica A: Statistical Mechanics and its Applications*, 314(1-4):25–34.

[8] Garey, M. e Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman & Co., New York.

[9] Garfield, E. (1979). *Citation indexing - its theory and application in Science, Technology and Humanities*. New York: John Wiley and Sons.

[10] He, S. e Spink, A. (2002). A comparison of foreign authorship distribution in jasist and the journal of documentation. *J. Am. Soc. Inf. Sci. Technol.*, 53(11):953–959.

[11] Huan, J.; Wang, W. e Prins, J. (2003). Efficient mining of frequent subgraphs in the presence of isomorphism. *icdm*, 00:549.

[12] Inokuchi, A.; Washio, T. e Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 13–23, London, UK. Springer-Verlag.

[13] Ketkar, N. S.; Holder, L. B. e Cook, D. J. (2005). Subdue: compression-based frequent pattern discovery in graph data. In *OSDM '05: Proceedings of the 1st international workshop on open source data mining*, pp. 71–76, New York, NY, USA. ACM Press.

[14] Kuramochi, M. e Karypis, G. (2004). An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1038–1051.

[15] Leydesdorff, L. e Wagner, C. (2005). Mapping global science using international co-authorships: a comparison of 1990 and 2000.

[16] Liu, X.; Bollen, J.; Nelson, M. L. e de Sompel, H. V. (2005). Co-authorship networks in the digital library research community. *Inf. Process. Manage.*, 41(6):1462–1480.

[17] Matsuda, T.; Horiuchi, T.; Motoda, H. e Washio, T. (2000). Extension of graph-based induction for general graph structured data. In *PADKK '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pp. 420–431, London, UK. Springer-Verlag.

[18] Nascimento, M. A.; Sander, J. e Pound, J. (2003). Analysis of sigmod's co-authorship graph. *SIGMOD Rec.*, 32(3):8–10.

[19] Newman, M. E. (2001a). Scientific collaboration networks. i. network construction and fundamental results. *Phys Rev E Stat Nonlin Soft Matter Phys*, 64(1 Pt 2).

[20] Newman, M. E. (2001b). Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys Rev E Stat Nonlin Soft Matter Phys*, 64(1 Pt 2).

[21] Nijssen, S. e Kok, J. N. (2004). A quickstart in frequent structure mining can make a difference. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 647–652, New York, NY, USA. ACM Press.

[22] Otte, E. e Rousseau, R. (2002). Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science*, 28(6):441–453.

[23] Raedt, L. D. e Kramer, S. (2001). The levelwise version space algorithm and its application to molecular fragment finding. In *IJCAI*, pp. 853–862.

[24] Smeaton, A. F.; Keogh, G.; Gurrin, C.; McDonald, K. e Sodring, T. (2003). Analysis of papers from twenty-five years of sigir conferences: what have we been doing for the last quarter of a century? *SIGIR Forum*, 37(1):49–53.

[25] Yan, X. e Han, J. (2002). gspan: Graph-based substructure pattern mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, p. 721, Washington, DC, USA. IEEE Computer Society.

[26] Yan, X. e Han, J. (2003). Closegraph: mining closed frequent graph patterns. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 286–295, New York, NY, USA. ACM Press.