PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Programa de Pós-Graduação em Informática

PARALELIZAÇÃO DO ALGORITMO NEXTCLOSURE NA MANIPULAÇÃO DE CONTEXTOS DE ALTA DIMENSIONALIDADE NO NÚMERO DE OBJETOS

Nilander Ricardo M. de Moraes

Belo Horizonte

2014

Nilander Ricardo M. de Moraes

PARALELIZAÇÃO DO ALGORITMO NEXTCLOSURE NA MANIPULAÇÃO DE CONTEXTOS DE ALTA DIMENSIONALIDADE NO NÚMERO DE OBJETOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática como requisito parcial para qualificação ao Grau de Mestre em Informática pela Pontifícia Universidade Católica de Minas Gerais.

Orientador: Luis E. Zárate Galvez

Co-orientador: Henrique Cota de Freitas

Belo Horizonte

2014

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Moraes, Nilander Ricardo M. de
M827p Paralelização do algoritmo nextclosure na manipulação de contextos de alta dimensionalidade no número de objetos / Nilander Ricardo M. de Moraes. Belo Horizonte, 2014.
68f.: il.

Orientador: Luis Enrique Zárate
Co-orientador: Henrique Cota de Freitas
Dissertação (Mestrado)- Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática.

1. Programação paralela (Computação). 2. Mineração de dados (Computação). 3. Algoritmos de computador. 4. Sistemas de recuperação da informação. I. Zárate, Luis Enrique. II. Freitas, Henrique Cota de. III. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática. IV. Título.

CDU: 681.3.01

Nilander Ricardo M. de Moraes

Paralelização do Algoritmo NextClosure na Manipulação de Contextos de Alta Dimensionalidade no Número de Objetos

Dissertação apresentada ao Programa de Pós-Graduação em Informática como requisito parcial para qualificação ao Grau de Mestre em Informática pela Pontifícia Universidade Católica de Minas Gerais.

Luis E. Zárate Galvez (Orientador) – PUC Minas

Henrique Cota de Freitas (Co-orientador) – PUC Minas

Humberto Torres Marques Neto – PUC Minas

Fabrício Benevenuto – Universidade Federal de Minas Gerais

Luis Fabrício Wanderley Góez – PUC Minas

Belo Horizonte, 25 de Abril de 2014.

AGRADECIMENTOS

À minha família, pelo apoio incondicional e pelo incentivo constante nessa mais recente etapa de minha vida. Obrigado pela compreensão a despeito de minhas mudanças de humor.

Agradeço ao prof. Dr. Zárate por ter me apresentado a essa área pouco conhecida nacionalmente que é a Análise Formal de Conceitos. Por acreditar em minha capacidade, jamais subestimando-me. Por seu otimismo e sua confiança de que o projeto daria certo, apesar dos inúmeros contratempos.

Ao prof. Dr. Henrique, que aceitou a co-orientação deste projeto. Seus comentários pontuais permitiram-me sair de situações labirínticas.

Ao meu caro amigo, Sérgio M. Dias, notório conhecedor da Análise Formal de Conceitos, sempre pronto a responder e a refletir sobre as mais diversas indagações concernentes à área. Suas críticas e sugestões ajudaram-me sobremaneira a finalizar com êxito este trabalho.

Por fim, agradeço também à CAPES pelo amparo financeiro.

RESUMO

O presente trabalho aborda o problema da manipulação de contextos densos e de alta dimensionalidade em número de objetos, o qual é um problema ainda em aberto na análise formal de conceitos. É investigada a geração da base mínima de implicações em contextos com tais características, onde o algoritmo *NextClosure* é empregado na obtenção de regras. Assim, este trabalho faz uso de computação paralela como forma de reduzir os tempos proibitivos observados nos cenários em que o contexto de entrada apresenta elevada densidade e alta dimensionalidade. São implementadas as versões sequencial e paralela do algoritmo *NextClosure* aplicado à geração de implicações. Os experimentos revelam uma redução de aproxidamente 70% no tempo de execução do contexto de maior dimensão e densidade, o que a atesta a viabilidade da estratégia aqui apresentada.

Palavras-chave: AFC. Implicações. Base Duquenne-Guigues.
Base Canônica. *NextClosure*. Reticulado Conceitual.

ABSTRACT

This work addresses the problem of handling dense contexts of high dimensionality in the number of objects, which is still an open problem in formal concept analysis. The generation of minimal implication basis in contexts with such characteristics is investigated, where the *NextClosure* algorithm is employed in obtaining the rules. Therefore, this work makes use of parallel computing as a means to reduce the prohibitive times observed in scenarios where the input context has high density and high dimensionality. The sequential and parallel versions of the *NextClosure* algorithm applied to generating implications are employed. The experiments show a reduction of approximately 70% in execution time in the contexts of greater size and density, which attests to the viability of the strategy presented in this work.

Keywords: FCA. Implications. Duquenne-Guigues base.
Stem base. *NextClosure*. Concept Lattice.

LISTA DE FIGURAS

FIGURA 1 Reticulado conceitual para um conjunto de plantas 21

FIGURA 2 Esquema de funcionamento do algoritmo 35

FIGURA 3 Nº de quasi-intensões em função da densidade 44

FIGURA 4 *Speedup* obtido com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 30% 50

FIGURA 5 *Speedup* obtido com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 50% 51

FIGURA 6 *Speedup* obtido com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 70% 52

FIGURA 7 Eficiência obtida com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 30% 53

FIGURA 8 Eficiência obtida com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 50% 54

FIGURA 9 Eficiência obtida com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 70% 54

FIGURA 10 Reticulado Conceitual para o contexto representando as sessões dos usuários ao *Orkut* 63

LISTA DE TABELAS

TABELA 1 Contexto para um conjunto de plantas 19

TABELA 2 Contextos sintéticos utilizados nas simulações 33

TABELA 3 Métricas para avaliação de desempenho 34

TABELA 4 Configuração do ambiente experimental 42

TABELA 5 Média do tempo sequencial (em segundos) para construção da base canônica dos contextos artificiais da Tabela 2 43

TABELA 6 Média do nº de regras obtido para cada grupo amostral 43

TABELA 7 Desvio-padrão do tempo de execução dos grupos amostrais na abordagem sequencial 45

TABELA 8 Desvio-padrão do nº de regras obtido para cada grupo amostral 46

TABELA 9 *Speedup* e Eficiência alcançados em DoublePrime com 8 *threads* de execução 46

TABELA 10 *Speedup* e Eficiência alcançados em LinClosure com 8 *threads* de execução 47

TABELA 11 Média do tempo paralelo (em segundos) para construção da base canônica dos contextos artificiais da Tabela 2 47

TABELA 12 *Speedup* alcançado com 8 *threads* e 8 núcleos disponíveis 49

TABELA 13 Simulações com 8 *threads* de execução 55

TABELA 14 Parte do Contexto Formal gerado a partir dos dados de acesso dos usários da rede social *Orkut* 58

LISTA DE ALGORITMOS

1 NextClosure 24

2 LinClosure 26

3 Duquenne-Guigues base mínima de implicações 26

4 Duquenne-Guigues (versão paralela) 38

5 Double Prime operador de derivação 38

6 Extent extensão do conceito 39

7 Intent intensão do conceito 40

8 LinClosure (versão paralela) 41

SUMÁRIO

1 INTRODUÇÃO **13**

1.1 Objetivos 16

1.2 Motivação 16

1.3 Justificativa 17

2 FUNDAMENTAÇÃO TEÓRICA **18**

2.1 AFC 18

2.1.1 Contexto Formal 18

2.1.2 Conceito Formal 19

2.1.3 Reticulado Conceitual 20

2.1.4 Regras de implicação 22

2.1.5 Base Duquenne-Guigues 24

3 TRABALHOS RELACIONADOS **27**

3.1 Processamento Paralelo e Distribuído Aplicado à AFC 27

3.2 AFC Aplicada a Redes Sociais 29

4 METODOLOGIA **31**

4.1 Considerações Iniciais 31

4.2 Geração de Contextos Formais Sintéticos 32

4.3 Métricas para Avaliação dos Experimentos 33

4.4 Algoritmo Paralelo 34

4.4.1 Definição de critérios de balanceamento de carga 36

4.4.2 Implementação do algoritmo escalável 36

5 ANÁLISE DOS RESULTADOS **42**

5.1 Resultados Experimentais 42

5.2 Desafio de Dresden 55

6 ESTUDO DE CASO **57**

6.1 Rede Social *Orkut* 57

7 CONCLUSÕES E TRABALHOS FUTUROS **64**

REFERÊNCIAS **66**

1 INTRODUÇÃO

Um crescente avanço tecnológico tem sido observado nos últimos anos, o que tem permitido a coleta e o armazém de grandes volumes de dados. Decorrente desse avanço, hoje, na sociedade de informação, a interação com sistemas computacionais ubíquos começa a ser uma realidade presente no cotidiano das pessoas, onde dados são gerados, armazenados e manipulados em tempo real. Seja por meio de *smartphones*, *tablets* ou dispositivos computacionais correlatos.

Dentre as fontes de dados disponíveis, destaca-se a *World Wide Web*, ou simplismente, *Web*. Amplamente difundida, essa rede possibilita a pesquisa e o compartilhamento de informações e de conhecimento, permitindo ainda, a interação entre pessoas e organizações geograficamente distantes ao redor do globo. Estas características evidenciam, portanto, que a Web atua tanto como um repositório fértil para pesquisa quanto como um veículo para disseminação de informações.

De fato, vivencia-se um período no qual a Internet passou a ter papel decisivo no ramo de negócios de organizações. Grandes expoentes tais como *Google*, *Yahoo!*, *Facebook*, *Twitter*, dentre outros empregam esse canal de comunicação como principal ativo em seu ramo de negócios. Não obstante, o cidadão comum também faz uso dessa plataforma para uma miríade de atividades, que vão desde a compra de produtos em lojas virtuais e pesquisa de informações até comunicação pessoal (*i.e. chat*). Dessa forma, o grande volume e a diversidade dos dados contidos nessa plataforma corroboram sua importância.

Certamente, a *Web* é a maior fonte de dados publicamente disponível (GOMIDE, 2012), motivo pelo qual despertou especial interesse da comunidade científica no tocante a obtenção de conhecimento dessa plataforma. Sabe-se que hoje, embora haja imensos repositórios de armazenamento de dados, em certas situações o ser humano se vê desprovido de informações e, por conseguinte, sem conhecimento para tomar decisões adequadas (SILVA, 2007). Isto ocorre porque as bases de dados são tão grandes que, em certa medida, acrestam dificuldade ao processo de recuperação de informação nesses repositórios, tornando tal processo proibitivo nesses casos (DIAS, 2010).

Nesse sentido, diversas técnicas para mineração de dados têm sido propostas para efetiva aquisição de conhecimento desses repositórios (GANTER; WILLE, 1997). Em meio

a essas técnicas destaca-se a Análise Formal de Conceitos (AFC), técnica esta capaz de realçar as relações estruturais dos dados em estudo (CARPINETO; ROMANO, 2004).

A assim chamada AFC envolve a representação hierárquica de Conceitos a partir de diagramas de linha específicos, também conhecidos como Reticulados Conceituais (GANTER; WILLE, 1997). Na AFC comumente um domínio de problema é modelado como uma tabela cruzada (Contexto Formal), que possibilita a representação das relações existentes entre o conjunto de instâncias em estudo (objetos) e as características que as descrevem (atributos). Além da representação estrutural dos dados, o Reticulado Conceitual provê uma *framework* em que várias técnicas de aquisição de conhecimento e análise de dados podem ser formuladas (KUZNETSOV; OBIEDKOV, 2008; CARPINETO; ROMANO, 2004).

De certo, o método formal da AFC para análise de dados já foi empregado em uma diversidade de áreas de conhecimento, tais como ciência da informação, psicologia, inteligência articial, biologia e linguística (PRISS, 2006). Sobretudo, trabalhos como o exposto em (KAYTOUE et al., 2011), em que são abordadas técnicas para mineração de dados de expressões genéticas; como os de Priss e de Neuhaus e Zimmermann, que tratam da aplicabilidade da AFC na área de segurança da informação (PRISS, 2011; NEUHAUS; ZIMMERMANN, 2009); e (PRISS, 2013; PRISS; RIEGLER; JENSEN, 2012) que empregam a AFC no modelo e análise das dificuldades conceituais encontradas no processo de aprendizagem de alunos; atestam a variada gama de aplicações da AFC e evidenciam seu grande potencial.

Não obstante, a AFC tem se mostrado como uma ferramenta eficaz para a comunidade de redes sociais (ROME; HARALICK, 2005; CUVELIER; AUFAURE, 2011; AUFAURE; GRAND, 2013), permitindo a identificação de comunidades e a análise e caracterização de atores e de conteúdo. Em parte, esse grande sucesso se deve, principalmente, a capacidade da AFC de representar graficamente as relações hierárquicas existentes entre o conjunto de dados sendo analisado. Contudo, um ponto digno de nota, diz respeito a capacidade de manipulação de bases relativamente grandes.

Sabe-se que a medida que o número de marcações e as dimensões do contexto aumentam, o esforço envolvido para análise da entrada pode aumentar drasticamente, chegando a casos em que o esforço computacional pode chegar a ser exponencial. Em decorrência disso, construir o Reticulado Conceitual pode não ser possível nessas circunstâncias. Ademais, quando construído, a visualização e interpretação do mesmo pode mostrar-se como um desafio, em razão da elevada quantidade de Conceitos.

Dessa forma, técnicas capazes de reduzir a complexidade de construção de Reticu-

lados, selecionando-se Conceitos específicos (KUZNETSOV; OBIEDKOV; ROTH, 2007; JAY; KOHLER; NAPOLI, 2008) ou mesmo atuando diretamente sobre o Contexto de entrada (DIAS; VIEIRA, 2010; RIMSA; ZáRATE; SONG, 2009), têm sido abordadas recentemente. Todavia, conforme observado pelos autores, tais simplificações podem ocultar dependências não frequentes, porém relevantes, prejudicando assim, a obtenção de conhecimento. Nesse sentido, o emprego de regras de implicação se mostra como uma alternativa interessante, por expressarem correlações e por capturarem padrões nos dados.

Mais especificamente, regras de implicação permitem a identificação de padrões pouco frequentes nos dados, ou seja, regularidades que ocorrem em uma quantidade pequena de registros (objetos). Embora sejam menos utilizadas para análise de dados (VIMIEIRO, 2007), as implicações são de interesse prático, pois podem explicitar novidades pouco conhecidas sobre os dados em estudo (COHEN et al., 2001). Daí a importância de tais estruturas.

Contudo, assim como na geração de Reticulados, gerar o conjunto completo de implicações também pode apresentar comportamento exponencial, no pior caso (GANTER, 2002). Nesse sentido, o emprego de arquiteturas paralelas mostra-se como alternativa viável na redução do tempo de execução.

Propostas e fundamentalismo matemático para soluções paralelas/distribuídas já foram apresentados por diversos autores (QI et al., 2004; HU et al., 2007; FU; NGUIFO, 2003). Entretanto, essas contribuições falham em avaliar o impacto da densidade do contexto de entrada no desempenho final da proposta. Além disso, carecem de informações sobre os recursos utilizados para a obtenção dos resultados, tais como as estruturas de dados utilizadas, a configuração do ambiente de teste, o número de computadores que compõem o aglomerado, a quantidade de núcleos envolvidos, dentre outras informações.

Os desafios de se gerar o conjunto completo de implicações de bases de dados relativamente grandes são conhecidos. De fato, como já mencionado, gerar o conjunto de implicações pode mostrar-se como um impedimento, em virtude do elevado custo computacional.

A literatura apresenta uma miríade de algoritmos para aquisição de bases de implicações. Dentre esses algoritmos, destaca-se o algoritmo *NextClosure* (GANTER, 2010) que, conforme sugerido por Guigues e Duquenne em (GUIGUES; DUQUENNE, 1986), pode ser empregado para a geração da base mínima de implicações (também denominada base canônica ou base *Duquenne-Guigues*). Sobretudo por fornecer o conjunto completo, mínimo e não-redundante de implicações, a base canônica tem papel especial por sumarizar

o conjunto de regras válidas. Em outras palavras, a base canônica corresponde ao subconjunto de implicações válidas (cobertura), de onde as demais regras podem ser obtidas por meio da combinação das regras desta base.

Portanto, neste trabalho o comportamento do algoritmo *NextClosure* na obtenção da base canônica de implicações é avaliado. Para tanto, foram implementadas as versões sequencial e paralela do algoritmo Duquenne-Guigues, em que o algoritmo *NextClosure* é utilizado na geração do conjunto regras. As duas abordagens são avaliadas e comparadas para se determinar em que cenários uma estratégia é mais adequada do que a outra, ou seja, em que situações o ganho alcançado com o paralelismo é significativo. Também é apresentado um estudo de caso da rede social *Orkut*, com dados do ano de 2009.

O restante desta dissertação esta organizado em 6 capítulos. O Capítulo 2 apresenta os fundamentos teóricos da AFC. O Capítulo 3 mostra os trabalhos relacionados a esta dissertação e apresenta as contribuições atingidas pelo presente trabalho. O Capítulo 4 expõe a metodologia empregada nos experimentos. No Capítulo 5 são apresentados os resultados empíricos. Por fim, conclusões e possíveis trabalhos futuros são apresentados no Capítulo 7.

1.1 Objetivos

O presente trabalho tem por objetivo o estudo, a implementação e a análise do algoritmo *NextClosure* para geração da base mínima de implicações. A escolha desse algoritmo se justifica no fato deste ser bem conceituado na comunidade científica, além é claro, de sua capacidade para o tratamento de contextos com alto grau de relações de incidência.

São também objetivos específicos deste trabalho, a análise desse algoritmo em sistemas computacionais sequencias e paralelos; o estudo do comportamento apresentado em cenários onde o contexto formal de entrada apresente elevada quantidade de relações (densidade) e alta dimensionalidade em número de objetos (linhas).

1.2 Motivação

Sabe-se que o problema do tratamento de contextos densos e de alta dimensionalidade é um problema que ainda se encontra em aberto, e que o mesmo é um dos fatores limitantes ao grande potencial da AFC. Um exemplo recorrente na litetura é o desafio exposto na ICFCA do ano de 2006, em Dresden, onde um contexto da ordem

de 120.000 × 70.000 objetos-atributos foi mencionado. De modo que esforços têm sido norteados no desenvolvimento de estratégias e algoritmos eficientes que consigam lidar em tempo hábil com bases de dados com essas características.

Não obstante, um outro problema recorrente na literatura é a inexistência de padronização para avaliação de algoritmos na área (DIAS, 2010). Seja pela ausência de exposição de métricas ou mesmo pelo uso inadequado de instâncias de entrada, a ausência de padronização acarreta em dificuldade na avaliação prática de experimentos empíricos. Dessa forma, a metodologia sugerida neste trabalho, apesar de não ser a principal contribuição nem o enfoque desta dissertação, é de grande serventia para a comunidade, pois apresenta métricas além daquelas conhecidas na área.

1.3 Justificativa

Como mencionado, a AFC possui uma enorme gama de aplicações, contudo a alta complexidade envolvida em situações onde a entrada apresenta elevado número de objetos, de atributos e de marcações (incidências na relação objeto × atributo), se mostra como um grande fator limitante. Seja pelo tempo de execução proibitivo ou mesmo pela capacidade de armazenamento, essa situação expõe a necessidade de aprimoramento das técnicas para análise de dados já existentes. Nesse sentido, o uso de estratégias paralelas/distribuídas, ainda que não resolva essa deficiência de forma definitiva, se revela como uma alternativa para se alcançar melhores resultados.

Ademais, sendo o problema do tratamento de contextos formais densos e de alta dimensionalidade um problema ainda em aberto e foco deste trabalho, espera-se por meio da exposição de métricas e da análise dos experimentos, um melhor entendimento do comportamento do algoritmo *NextClosure* aplicado a geração da base mínima de implicações. Espera-se também, através dos resultados, mostrar a viabilidade do uso de arquiteturas paralelas como forma de reduzir o alto tempo de execução observado quando o contexto de entrada apresentar seu pior caso (*i.e.* alta dimensionalidade e elevada densidade).

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os fundamentados teóricos da AFC. Um estudo mais aprofundado sobre o assunto pode ser encontrado em (CARPINETO; ROMANO, 2004) ou em (GANTER; WILLE, 1997).

2.1 AFC

A Análise Formal de Conceitos é um ramo da matemática aplicada baseada na matematização da noção de conceito e na estruturação desses conceitos em uma ordem hierárquica (reticulado conceitual). Sua formalização teve início com o trabalho de Wille (WILLE, 1982), que na década de 80 propôs considerar cada elemento do reticulado como um conceito. Wille tinha por meta estabelecer uma linguagem comum que pudesse ser empregada tanto por especialistas quanto por não-especialistas na área de reticulados.

Dado sua simplicidade e vasta gama de aplicações, a AFC tem despertado crescente interessse por parte da comunidade científica. De fato, a AFC vem se tornando a principal técnica utilizada para o processamento de dados e conhecimento (CARPINETO; ROMANO, 2004). Todavia, gerar o conjunto completo de conceitos formais e ordena-los hierarquicamente apresenta comportamento exponencial, no pior caso.

Apesar de na prática esse comportamento ser raramente observado, há enorme interesse da comunidade em apresentar soluções eficientes, uma vez que essa restrição esta intimamente relacionada com o problema do tratamento de contextos densos e de alta dimensionalidade. Problema este que ainda se encontra em aberto (GÉLY et al., 2005; GÉLY; MEDINA; NOURINE, 2010).

2.1.1 Contexto Formal

Um Contexto Formal (G, M, I) consiste em dois conjuntos G e M, e uma relação binária I entre esses conjuntos. Os elementos de G são denominados objetos, enquanto que os de M são denominados atributos. Caso um objeto g possua uma relação I com um atributo m, essa relação é expressa por gIm ou $(g, m) \in I$. O que é interpretado como "objeto g possui o atributo m".

Na Tabela 1, um exemplo de contexto formal para um conjunto de plantas é apresentado. Nesse contexto, objetos denotam plantas e atributos correspondem às características dessas plantas. Células marcadas com '×' denotam a presença da respectiva característica, ou, de forma similar, que a característica correspondente descreve aquele objeto.

Tabela 1: Contexto para um conjunto de plantas

	Precisa de água	Clima			Tem Frutos	Tem flores
		tropical	temperado	inverno		
cana	×	×				
eucalipto	×	×	×	×	×	×
sucupira	×	×	×		×	×
uveira	×			×	×	×
roseira	×		×		×	×
bananeira	×	×			×	×

Para um conjunto $A \subseteq G$ de objetos, A' é definido como o conjunto de atributos comuns aos objetos em A. E matematicamente, é expresso como:

$$A' := \{m \in M \mid gIm \; \forall g \in A\}. \tag{2.1}$$

Em correspondência, para o conjunto $B \subseteq M$ de atributos, B' é definido como o conjunto de objetos com todos os atributos em B. Formalmente, tem-se:

$$B' := \{g \in G \mid gIm \; \forall m \in B\}. \tag{2.2}$$

2.1.2 Conceito Formal

Um Conceito Formal de um contexto (G, M, I) consiste em um par ordenado (A, B), onde a seguinte propriedade se aplica

$$A \subseteq G, \; B \subseteq M, \; A' = B \; e \; B' = A. \tag{2.3}$$

O subconjunto A de objetos é denominado extensão, enquanto que o subconjunto B de atributos é denotado intensão. Assim, um conceito é caracterizado por sua extensão e intensão. A extensão corresponde a todos os objetos pertencentes ao conceito enquanto que a intensão possui todos os atributos compartilhados por esses objetos.

Uma consequência imediata à definição de conceito é que nem todo subconjunto de objetos (extensão) ou de atributos (intensão) podem constitiuir um Conceito Formal. Um subconjunto A de G é extensão se, e somente se, $A'' = A$ (GANTER; WILLE, 1997);

assim, neste caso o único conceito em que A seria extensão seria (A, A'). Isto também é válido para o conjunto de atributos.

Como exemplo pode-se citar os objetos *(roseira, sucupira, eucalipto)*. Tal conjunto de objetos constitui uma extensão do conceito, pois *(roseira, sucupira, eucalipto)'* deriva em *(agua, temperado, frutos, flores)* que por sua vez deriva em *(roseira, sucupira, eucalipto)*. Em outras palavras, *(roseira, sucupira, eucalipto)* é extensão do conceito pois aplicando-se o operador de derivação duas vezes sobre esse conjunto, obtém-se um conjunto idêntico ao conjunto de entrada. Por outro lado, *(uveira, cana)* não é uma extensão, pois *(uveira, cana)''* não mapeia para o mesmo conjunto de objetos de entrada.

O conjunto de conceitos de um contexto também possibilita uma interpretação geométrica, uma vez que o mesmo pode ser visto como um retângulo máximo da tabela representando o contexto (CARPINETO; ROMANO, 2004). Formalmente, um retângulo máximo de um contexto (G, M, I) é um par (A, B) onde $A \subseteq G$ e $B \subseteq M$, tal que

$$\forall g \in G \backslash A, \exists m \in M | (g, m) \notin I, \tag{2.4}$$

$$\forall m \in M \backslash B, \exists g \in G | (g, m) \notin I. \tag{2.5}$$

Essas condições requerem que para cada objeto não pertencente ao retângulo máximo, exista ao menos um atributo do retângulo máximo que não é compartilhado pelo objeto, e, em correspondência, que para todo atributo não incluído no retângulo máximo, exista ao menos um objeto do retângulo máximo que não compartilhe o atributo. Por exemplo, tomando o contexto da Tabela 1, *(a, b, 1, 2)* é um retângulo máximo, enquanto que *(c,d, 3, 4)* não o é, pois o mesmo viola a segunda propriedade.

2.1.3 Reticulado Conceitual

Uma relação de ordem no conjunto de conceitos de um contexto formal é definido da forma a seguir. Se $(A1, B1)$ e $(A2, B2)$ são conceitos em $\beta(G, M, I)$, diz-se que $(A1, B1)$ é um subconceito de $(A2, B2)$, ou que $(A2, B2)$ é um superconceito de $(A1, B1)$, se $A1 \subseteq A2$. Isto pode ser representado por $(A1, B1) \leq (A2, B2)$. Neste caso, pode-se dizer que $(A1, B1)$ é menor ou mais específico do que $(A2, B2)$, ou que $(A2, B2)$ é maior ou mais geral do que $(A1, B1)$. A relação $\leq$ é denominada ordem hierárquica ou simplesmente ordem dos conceitos.

Assim, ao conjunto de todos os conceitos $\beta(G, M, I; \leq)$ ordenados segundo a ordem hierárquica é denominado Reticulado de Conceitos ou Reticulado Conceitual. Essa estrutura é formalmente descrita pelo teorema a seguir.

Teorema 2.1. ***Teorema básico do Reticulado de Conceitos.*** *Se* (G, M, I) *é um contexto formal, então* $\beta(G, M, I; \leq)$ *é um reticulado completo dado por:*

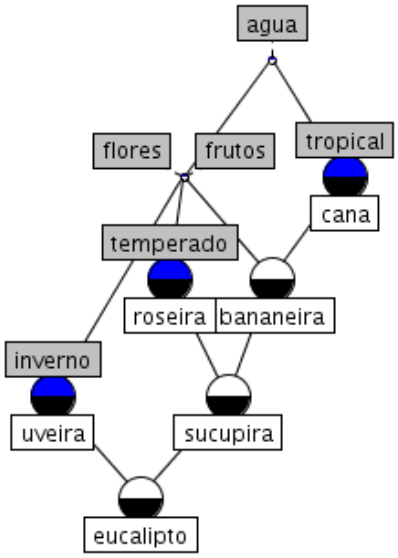
$$\vee_{j \in J}(A_j, B_j) = ((\cup_{j \in J} A_j)'', \cap_{j \in J} B_j), \tag{2.6}$$

$$\wedge_{j \in J}(A_j, B_j) = (\cap_{j \in J} A_j, (\cup_{j \in J} B_j)''). \tag{2.7}$$

A prova da correção do Teorema 2.1 é fornecida por Carpineto e Romano em (CARPINETO; ROMANO, 2004).

O reticulado correspondente ao conjunto de plantas apresentado na Tabela 1 é mostrado a seguir, na Figura 1. Esse reticulado contém um total de 8 de conceitos, incluindo o topo (conjunto de atributos compartilhados por todas as plantas) e a base da hierarquia (conjunto de plantas que compartilham todos os atributos).

Figura 1: Reticulado de conceitos para o conjunto de plantas da Tabela 1



É importante ressaltar que na Figura 1 apenas uma fração do conjunto de possíveis combinações entre plantas e suas respectivas características são representadas, pois so-

mente os pares que são completos de acordo com a relação de incidência são representados no reticulado. Por exemplo, não existe par em que o atributo 'agua' não esteja presente, pois toda planta precisa de água.

Por outro lado, é visualmente perceptível que as similaridades e diversidades do conjunto de plantas mostradas no reticulado estão bem representadas. Por exemplo, o fato do atributo 'agua' estar localizado no topo dessa estrutura, indica que qualquer elemento inferior, possuirá essa característica.

2.1.4 Regras de implicação

Diz-se que um contexto (G, M, I) satisfaz uma implicação $Q \rightarrow R$, com $Q, R \subseteq M$, se

$$\forall g \in G,\ gIq \text{ para todo } q \in Q \text{ implica } gIr \text{ para todo } r \in R.$$

Ou seja, uma regra de implicação entre dois conjuntos Q e R é válida para um determinado contexto, se o conjunto de objetos descrito pelos atributos em Q for também descrito pelos atributos em R. De forma análoga, uma regra de implicação pode ser interpretada como "todos os objetos que possuem os atributos de Q também possuem os atributos de R". Os conjuntos Q e R são denominados, respectivamente, premissa e conclusão.

Embora seja possível obter o conjunto completo de implicações que um contexto satisfaça, uma vez que o número de subconjuntos de M é finito, esta tarefa se revela impraticável na maioria das vezes(CARPINETO; ROMANO, 2004). Isto porque muitas das regras geradas são triviais ou redundantes.

Assim, uma forma mais conveniente de se obter o conjunto de implicações de um contexto seria a partir de um conjunto de regras informativas, das quais as demais regras pudessem ser obtidas. De fato, mecanismos de inferência exploram justamente esta característica.

Mecanismos de inferência permitem, por meio de sucessivas aplicações de suas regras, gerar o conjunto completo de implicações a partir de um subconjunto de implicações. Como exemplo de tal mecanismo, pode-se citar os axiomas de Armstrong, apresentados a seguir.

Definição 2.1 (Axiomas de Armstrong)**.** Um conjunto $\mathcal{L}$ de implicações em M é fechado

se, e somente se, as seguintes condições forem satisfeitas para todo $Q, R, W, Z \subseteq \mathcal{L}$:

- $Q \rightarrow Q \in \mathcal{L}$,
- se $Q \rightarrow R \in \mathcal{L}$, então $Q \cup Z \rightarrow R \in \mathcal{L}$,
- se $Q \rightarrow R \in \mathcal{L}$ e $R \cup Z \rightarrow W \in \mathcal{L}$, então $Q \cup Z \rightarrow W \in \mathcal{L}$.

Dois conjuntos de implicações sobre M são ditos equivalentes se possuírem o mesmo fecho. Portanto, se $\mathcal{L}$ e $\mathcal{K}$ são equivalentes, então $\mathcal{K}$ é uma cobertura de $\mathcal{L}$. Logo, uma cobertura mínima de um conjunto de implicações $\mathcal{L}$ é o conjunto de menor tamanho entre os demais conjuntos equivalentes a $\mathcal{L}$. Um caso particular de cobertura mínima é o que usa a noção de conjunto pseudo-fechado, definido a seguir.

Definição 2.2 (Conjunto Pseudo-fechado)**.** Um conjunto $P \subseteq M$ é denominado **pseudo-fechado** se $P \neq P''$ e $Q'' \subseteq P$ para todo conjunto pseudo-fechado $Q \subseteq P$, $Q \neq P$.

Reformulando essa definição para contextos formais, tem-se

Definição 2.3 (Pseudo-intensão)**.** Dado um Contexto Formal (G, M, I) com M finito. Um subconjunto $P \subseteq M$ é uma **pseudo-intensão** de (G, M, I) se, e somente se

- P não é intensão do conceito, e
- se $Q \subset P$, então existe algum objeto $g \in G$ tal que $Q \subseteq g'$ mas $P \nsubseteq g'$.

O conjunto de implicações da forma $P \rightarrow P''$, onde P é pseudo-fechado é denominado base canônica ou base Duquenne-Guigues (GUIGUES; DUQUENNE, 1986). Essa base é garantidamente o conjunto de menor tamanho dentre os conjuntos equivalentes de $\mathcal{L}$. Ademais, tal cobertura possui a propriedade de ser não-redundante, isto é, não é possível remover implicações desse conjunto com o intuito de reduzir seu tamanho.

Na prática, uma versão um pouco mais simplificada da base canônica é utilizada, já que parte dos elementos presentes na premissa se repetem na conclusão. Esta simplificação não altera o sentido, nem incorre em perda de informação, e é expressa a seguir.

$$\{P \rightarrow P'' \backslash P \mid P \text{ é pseudo-fechado}\} \tag{2.8}$$

2.1.5 Base Duquenne-Guigues

A base mínima de implicações pode ser obtida com auxílio do algoritmo *NextClosure* proposto por Ganter em (GANTER, 2010) e extensivamente discutido em (GANTER; WILLE, 1997; GANTER, 2002). Tal algoritmo é capaz de encontrar conjuntos fechados, dado um operador de fecho, em ordem lexicográfica.

Dado um Contexto Formal (G, M, I), *NextClosure* funciona sobre a premissa de que existe uma ordem total entre os atributos de M. Mais especificamente, o algoritmo impõe a seguinte ordem sobre os elementos de M: $\{m_1 < m_2 < \ldots < m_{|M|-1} < m_{|M|}\}$.

Depois de imposta uma ordem linear sobre os elementos de M, cada subconjunto pertencente a esse conjunto é ordenado de acordo com uma ordem lexicográfica ($\leq$), definida como a seguir. Um subconjunto A de M é menor do que um subconjunto B de M, se o menor elemento que difere A de B pertencer a B. Por exemplo, $\{135\} \leq \{134\}$, pois o menor elemento que difere os dois conjuntos, pertence ao segundo conjunto (*i.e.* 4).

NextClosure é a apresentado a seguir no Algoritmo 1.

Algoritmo 1: NEXTCLOSURE

Entrada: Um operador de fecho $X \rightarrow X''$ sobre conjunto M
Saída: A é substituído pelo próx. conjunto fechado em ordem lexicográfica

```
1  i ← maior elemento de M
2  i ← succ(i)
3  sucesso = F
4  repita
5  |  i ← pred(i)
6  |  se i ∉ A então
7  |  |  A ← A ∪ {i}
8  |  |  B ← A''
9  |  |  se B\A não contém elemento < i então
10 |  |  |  A ← B
11 |  |  |  sucesso ← V
12 |  |  fim
13 |  senão
14 |  |  A ← A\{i}
15 |  fim
16 até sucesso ou i = menor elemento de M
17 retorna A
```

Conforme mostrado anteriormente, a base mínima de implicações é constituída pelo conjunto de regras da forma $P \rightarrow P''$, em que P é pseudo-fechado. As definições

seguintes fazem uso desta característica, e descrevem o mecânismo com o qual o conjunto de implicações pode ser obtido.

Definição 2.4 (Conjunto Quasi-fechado)**.** Um conjunto $Q \subseteq M$ é **quasi-fechado** se este contiver o fecho de todo conjunto quasi-fechado contido em Q.

Em outras palavras, Q é quasi-fechado se, e somente se, para cada $Q_0 \subset Q$ com $Q_0 \neq Q$, $Q_0'' \subseteq Q$.

Proposição 2.1 (Conjunto Quasi-fechado)**.** Um conjunto é **quasi-fechado** se, e somente se, o mesmo for fechado ou pseudo-fechado.

Como consequência imediata da Definição 2.4, tem-se a seguinte constatação acerca de conjuntos quasi-fechados.

Proposição 2.2. Q é quasi-fechado se, e somente se, Q satisfaz a seguinte condição: se $P \subset Q$, $P \neq Q$, é pseudo-fechado, então $P'' \subseteq Q$.

A proposição anterior mostra como o fecho de quasi-conjuntos pode ser calculado para quaisquer conjuntos $Q \subseteq M$. Matematicamente, o operador de fecho ϕ é definido como a seguir, considerando $\mathcal{L}$ uma base canônica e $X \subseteq M$:

$$\phi(X) := X \cup \bigcup \{P'' | P \to P'' \in \mathcal{L}, P \subset X, P \neq X\}.$$

Esse operador de fecho é aplicado sucessivas vezes sobre X, formando $\phi(X) = \phi(\phi(\phi(\ldots\phi(X))))$, até que $\phi(X) = \phi(\phi(X))$ seja obtido. O comportamento do operador ϕ é descrito pelo Algoritmo 2.

Diante das definições anteriores é fácil definir um algoritmo para computar a base mínima de implicações. O Algoritmo 3 gera todo o conjunto pseudo-fechado para um dado operador de fecho. Mais especificamente, o algoritmo *NextClosure* é utilizado com o auxílio do operador de fecho de conjuntos quasi-fechados. Isto gera o conjunto quasi-fechado em ordem lexicográfica, mas o algoritmo só registra aqueles que não são fechados, o que resulta na lista completa de conjuntos pseudo-fechados.

Algoritmo 2: LinClosure

Entrada: Uma lista $\mathcal{L} \leftarrow [\mathcal{L}[1], \ldots, \mathcal{L}[n]]$ de implicações em M e um conjunto $X \subseteq M$
Saída: Fecho $\mathcal{L}(X)$ de X

para todo $x \in M$ **faça**
  $avoid[x] \leftarrow \{1, \ldots, n\}$
  para $i \leftarrow 1$ ***até*** n **faça**
    $A \rightarrow B \leftarrow \mathcal{L}[i]$
    se $x \in A$ **então**
      $avoid[x] \leftarrow avoid[x] \setminus \{i\}$
    fim
  fim
fim
$used\_imps \leftarrow \emptyset$
$old\_closure \leftarrow \{-1\}$
$new\_closure \leftarrow X$
enquanto $new\_closure \neq old\_closure$ **faça**
  $old\_closure \leftarrow new\_closure$
  $T \leftarrow M \backslash new\_closure$
  $usable\_imps \leftarrow \bigcap_{x \in T} avoid[x] \cap \bigcup_{x \in new\_closure} avoid[x]$
  $use\_now\_imps \leftarrow usable\_imps \setminus used\_imps$
  $used\_imps \leftarrow usable\_imps$
  para todo $i \in use\_now\_imps$ ***com*** $A \rightarrow B \leftarrow \mathcal{L}[i]$ **faça**
    $new\_closure \leftarrow new\_closure \cup B$
  fim
fim
$\mathcal{L}(x) \leftarrow new\_closure$
retorna $\mathcal{L}(x)$

Algoritmo 3: Duquenne-Guigues base mínima de implicações

Entrada: Contexto Formal $\mathcal{K}(G, M, I)$
Saída: A base canônica $\mathcal{L}$

$\mathcal{L} \leftarrow \emptyset$
$A \leftarrow \emptyset$
enquanto $A \neq M$ **faça**
  se $A \neq A''$ **então**
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{A \rightarrow A''\}$
  fim
  $A \leftarrow$ `next_closure(`A`,`Φ`)`
fim
retorna $\mathcal{L}$

3 TRABALHOS RELACIONADOS

3.1 Processamento Paralelo e Distribuído Aplicado à AFC

Uma notável contribuição para a área da Análise Formal de Conceitos é apresentada por (QI et al., 2004), onde os autores propõem um novo algoritmo distribuído para extração de conceitos formais por meio do particionamento do espaço de busca de conceitos. Em tal trabalho, a abordagem de verificação da validade dos subespaços, isto é, dos subconjuntos que possuem maior probabilidade de gerar conceitos formais, é inédita.

Entretanto, naquele trabalho apenas o formalismo matemático é apresentado, o que o torna sob o ponto de vista prático, pouco impactante, pois não são apresentados os resultados empíricos da abordagem proposta.

Fu e Nguifo também apresentam um algoritmo para obtenção de conceitos a partir de espaços de busca de forma paralela (FU; NGUIFO, 2003). Em tal proposta, atributos redundantes são removidos do Contexto Formal de entrada antes da etapa de geração das partições, a fim de diminuir o esforço do algoritmo na extração dos conceitos formais. Outra característica importante a ser ressaltada sobre o algoritmo proposto pelos autores, é que o mesmo baseia-se no algoritmo *NextClosure* para a efetiva obtenção do conjunto de conceitos. Ademais, a abordagem adotada pelos autores não leva em consideração o balanceamento das cargas de trabalho, o que se revela como um dos fatores limitantes à estratégia empregada.

Naquele trabalho os autores concluem por meio dos resultados experimentais que, em decorrência da utilização do algoritmo *NextClosure* para obtenção dos conceitos, a proposta paralela é capaz de lidar com contextos de entrada de alta dimensionalidade, mas que em decorrência do não balanceamento das cargas de trabalho, o algoritmo pode apresentar variações bruscas de desempenho dependendo do fator de carga escolhido (*i.e.* DP).

Em contraponto aos resultados do trabalho de (FU; NGUIFO, 2003), (MORAES; ZÁRATE; FREITAS, 2010) apresentam uma estratégia para geração de conceitos formais que leva em consideração o balaceamento das cargas de trabalho. De fato, tal característica é uma vantagem em relação a abordagem de (FU; NGUIFO, 2003), pois como ressaltado

pelos autores, o tempo de execução pode ser estimado.

De forma similar, Krajca *et al* também exploram a paralelização de algoritmos no processo de geração de conceitos em (KRAJCA; OUTRATA; VYCHODIL, 2008). De fato, a versão paralela discutida nesse trabalho baseia-se no algoritmo exposto em (VYCHODIL, 2008). Nesse trabalho a estratégia de geração de conceitos é muito semelhante ao algoritmo *CbO* de Kuznetsov (KUZNETSOV, 1999), diferindo-se somente na ordem dos conceitos gerados.

Em (KRAJCA; OUTRATA; VYCHODIL, 2008) duas versões (sequencial e paralela) de um mesmo algoritmo são apresentadas. Os autores fazem uma breve explicação acerca da técnica utilizada para obtenção de conceitos, e atestam a viabilidade do emprego de busca em profundidade como estratégia para a não geração de conceitos repetidos. Além disso, realizam testes experimentais comparativos entre a versão sequencial e a paralela, bem como com outros algoritmos da literatura (GANTER, 2010; LINDIG; GBR, 2000; BERRY; BORDAT; SIGAYRET, 2007). Todavia, os autores não mencionam quais estruturas de dados foram utilizadas na implementação desses algoritmos.

Por sua vez, (HU et al., 2007) utilizam uma abordagem de particionamento do Contexto Formal de entrada, o que se difere dos trabalhos mencionados anteriormente, pois nessa estratégia o contexto é divido ou horizontalmente ou verticalmente, e como resultado dessa operação obtem-se subcontextos, ou seja, contextos formais parciais. De posse desses subcontextos, os subconceitos são então extraídos e unidos por meio de um operador específico de união, a fim de se encontrar o conjunto final de conceitos formais.

Essa estratégia possui a vantagem de diminuir consideravelmente o grau de complexidade demandado para extrair o conjunto de relações entre objetos e atributos. No entanto, esse ganho é praticamente descartado no processo de união dos subconceitos. De fato, num primeiro momento, dividir o contexto parece a escolha óbvia a ser feita para se reduzir a complexidade de geração dos conceitos. Todavia, assim como gerar conceitos, uni-los também apresenta ordem exponencial.

Em (VALTCHEV; DUQUENNE, 2003), é apresentada uma estratégia de divisão e conquista para construção do Reticulado de Conceitos. Em tal estratégia, o Reticulado é construído a partir de reticulados parciais que, tomados dois a dois por meio do produto cartesiano, formam o Reticulado completo. Naquele trabalho, os autores apresentam uma nova técnica para identificação de nós inválidos, que consiste em calcular, juntamente com o Reticulado, o conjunto de regras de implicação dos Reticulados parciais.

Valtchev e Duquenne afirmam que essa técnica é eficaz, uma vez que possibilita a rápida identificação de nós inválidos, isto é, o produto cartesiano de Reticulados parciais que não constituem um Conceito. Os autores também afirmam que o método proposto apresenta melhor desempenho do que o algoritmo *NextClosure*, entretanto não evidenciam esse comportamento de forma experimental.

Finalmente, (LI et al., 2003) por sua vez apresentam todo o formalismo matemático necessário para a efetiva geração de conceitos formais em sistemas distribuídos, assegurando assim a viabilidade de sua utilização. Embora naquele trabalho não sejam discutidos os aspectos práticos do emprego de arquiteturas paralelas.

O presente trabalho diferencia-se dos trabalhos anteriores, por apresentar as estruturas de dados utilizadas na implementação dos algoritmos e por detalhar a metodologia sugerida e o ambiente onde as simulações foram realizadas. Ademais, são expostas algumas métricas específicas da área de computação paralela e distribuída, tais como o *speed up* e a *eficiência*, muito comuns na avaliação de algoritmos, porém pouco discutidas no campo da AFC.

Assim, este trabalho concentra-se na paralelização do processo de geração de regras de implicação. Mais especificamente, na paralelização da base canônica de implicações. Apesar da geração do conjunto de implicações mostrar-se como exponencial, no pior caso, os experimentos conduzidos com a versão paralela revelam uma redução significativa no tempo de execução em comparação com a versão sequencial. Atestando dessa forma, a viabilidade da utilização de arquiteturas paralelas como método de auxílio a redução dos tempos proibitivos, verificados na manipulação de bases de larga escala.

3.2 AFC Aplicada a Redes Sociais

Em (POELMANS et al., 2010), um estudo acerca do estado da arte da Análise Formal de Conceitos é apresentado. Naquele trabalho são coletados 702 artigos da área, compreendidos no período de 2003 a 2009, de uma miríade de conferências bem conceituadas pela comunidade científica. A capacidade de visualização gráfica da AFC é empregada para se descobrir quais são os tópicos de maior interesse. Os autores destacam o crescente interesse da comunidade pelo melhoramento da escalabilidade de algoritmos da AFC no tratamento de bases de dados grandes e complexas, dos quais 5% dos artigos analisados se referem a esse tema.

Outro estudo similar ao exposto por Poelman *et al*, é apresentado em (DOERFEL;

JÄSCHKE; STUMME, 2012). Os autores realizam uma análise das comunidades formadas por autores das 3 principais conferências relacionadas a AFC, a saber: *ICFCA*, *ICCS* e *CLA*. Por meio do arcabouço fornecido pela própria AFC, aliado a teoria de grafos e com os algoritmos de classificação *PageRank* (BRIN; PAGE, 1998) e *HITS* (KLEINBERG, 1999), os autores identificam as publicações e os autores mais influentes nessas conferências.

Rome *et al* também fazem uso de Reticulados Conceituais para a identificação de comunidades na Web, em (ROME; HARALICK, 2005). Naquele trabalho, os autores modelam o Contexto Formal em termos de *Hubs* e *Authorities*, ou seja, páginas da Web com um número elevado de *links* para páginas relacionadas e páginas apontadas por muitas páginas, respectivamente. Os autores também definem uma comunidade da Web como sendo um subgrafo bipartido completo (*i.e.* diclique).

Segundo Rome *et al*, uma das vantagens decorrentes do mapeamento adotado, é que a identificação de comunidades corresponde diretamente ao conjunto de conceitos formais obtidos do contexto de entrada. Contudo, os autores mostram que em razão desse modelamento, a manipulação do contexto resultante pode mostrar-se como impraticável, em virtude de suas dimensões e densidade.

(CUVELIER; AUFAURE, 2011) investiga a identificação de alterações em informações repassadas no *Twitter*. Os autores propõem a ferramenta *EVARIST* que, dado um assunto, coleta os *tweets* relacionados aquele assunto. *EVARIST* remove as *stop words* e os sinais de pontuação, e monta um Contexto Formal em termos de *tweets* (objetos) e palavras que ocorrem naquele *tweet* (atributos). Feito isto, é construído o Reticulado Conceitual em cima do contexto gerado.

Tal como os autores discutem, o Reticulado propricia uma forma elegante e clara para visualizar as alterações ao longo dos *tweets*. Também possibilitando a identificação de como uma mesma informação é repassada. Porém, o Reticulado resultante pode apresentar elevada quantidade relações, o que incorreria em maior dificuldade para o entendimento do conjunto de relações. Nesse sentido, Cuvelier e Aufaure propõem a visualização dos conceitos mais importantes (aqueles com maior número de palavras).

4 METODOLOGIA

4.1 Considerações Iniciais

Os contextos formais utilizados para a realização das simulações do presente trabalho foram gerados de forma aleatória, com o intuito de se obter um cenário de testes mais confiável. Além disso, foram especificadas as densidades mínima, intermediárias e máxima para os contextos formais empregados nos experimentos, pois tal parâmetro esta diretamente relacionado ao desempenho dos algoritmos da AFC, tanto de geração de conceitos quanto de obtenção regras.

Para tanto, foi utilizada a ferramenta *SCGaz* (RIMSA; ZáRATE; SONG, 2012) para geração das cargas de trabalho. Essa ferramenta possibilita a especificação do número de objetos e de atributos para o contexto a ser gerado, bem como a especificação de uma densidade arbitrária, calculada com base nas dimensões adotadas para o contexto. Nesse ponto, é importante ressaltar que os contextos gerados pelo *SCGaz* são irredutíveis.

Visto que o alto desempenho computacional, a ser alcançado na obtenção das regras de implicação, também é um dos objetivos do presente trabalho, a implementação de ambas abordagens (versão sequencial e paralela) para obtenção de regras de implicação se deu em linguagem de programação *C*. Sendo que, para a abordagem paralela, foi empregada a biblioteca *OpenMP* para acesso aos recursos de paralelismo intra-nó (*i.e. multithreading*), uma vez que esta é o padrão *de facto* na área de computação paralela de alto desempenho.

Destaca-se ainda que as estruturas de dados adotadas para ambas abordagens foram as mesmas. Isto foi feito de modo a minimizar o surgimento de eventuais distorsões que poderiam afetar os resultados experimentais, em virtude da adoção de estruturas de dados divergentes.

O restante deste capítulo esta organizado da forma a seguir. A seção 4.2 apresenta como se deu processo de geração dos contextos sintéticos empregados nas simulações. Na seção 4.3 são expostas as métricas para avaliação dos experimentos. A seção 4.4 descreve a arquitetura paralela adotada, expondo o balanceamento de carga e as decisões de implementação da abordagem paralela.

4.2 Geração de Contextos Formais Sintéticos

Como mencionado anteriormente, de forma a garantir um ambiente controlado para os testes, os contextos formais empregados nas simulações foram gerados de forma aleatória, com controle de densidade e dimensão. A escolha de contextos sintéticos se justifica no fato de que o uso de bases reais pode especializar, em vez de generalizar a análise dos resultados, e necessitar de um processo de discretização, o que implicaria em mais um aspecto a ser considerado nos experimentos.

Para a geração dos contextos sintéticos, optou-se pela escolha de um conjunto de densidades que pudessem representar de forma adequada o comportamento do algoritmo em diversas situações. Dessa forma, foram empregadas para as simulações as respectivas densidades: mínima, 30%, 50%, 70% e máxima.

Como os contextos gerados pelo *SCGaz* possuem a característica de serem irredutíveis, as densidades mínima e máxima podem variar de acordo com a dimensionalidade especificada (*i.e.* nº de objetos e de atributos). O que pode tornar inconveniente comparar o comportamento de 2 (dois) ou mais contextos de dimensões diferentes nesses extremos, pois o parâmetro densidade será divergente em cada entrada. Assim, para possibilitar a comparação entre diferentes contextos, optou-se pelo uso de densidades intermediárias (*i.e.* 30%, 50% e 70%), tendo em vista a avaliação do algoritmo frente a variação de outros parâmetros.

Tendo em vista que dois ou mais contextos formais podem se diferenciar em suas matrizes de incidência mesmo apresentando características em comum, tais como dimensões e densidades, para as simulações serão empregadas amostras de tamanho 3 para cada contexto sintético considerado. Isto será feito a fim de se contemplar o impacto do conjunto de marcações (incidências) no desempenho de ambas abordagens. A Tabela 2 apresenta os contextos sintéticos empregados nos experimentos, onde cada linha representa um grupo amostral.

Nota-se nessa Tabela que o conjunto de contextos sintéticos considerados apresentam poucos atributos em relação a quantidade de objetos. Isto é justificado, pois o algoritmo para obtenção da base mínima de implicações possui pior caso no número de atributos do contexto de entrada. Ademais, há também o fato de que esta é a situação que ocorre com maior frequência em casos reais de aplicação da AFC. Portanto, de forma a reproduzir esse comportamento e levando em conta as limitações do algoritmo, foram adotados contextos que emulam as respectivas características.

Tabela 2: Contextos sintéticos utilizados nas simulações

$\|M\| \times \|G\|$	Densidade (%)				
	mín.	intermediária			máx.
15×1000	21,93	30	50	70	78,07
15×5000	29,88	30	50	70	70,12
15×10000	34,97	-	50	-	65,03
20×1000	13,85	30	50	70	86,15
20×5000	18,42	30	50	70	81,58
20×10000	21,11	30	50	70	78,88
25×1000	10,6	30	50	70	89,4
25×5000	13,62	30	50	70	86,38
25×10000	14,81	30	50	70	85,19

4.3 Métricas para Avaliação dos Experimentos

Como a AFC esta intimamente relacionada à área da matemática aplicada, vários trabalhos disponíveis na literatura são de autores que não são da área da computação. Isto por vezes acarreta em trabalhos cuja metodologia de avaliação não comtempla métricas cruciais para avaliação de desempenho das abordagens propostas, chegando a casos, em que apenas os aspectos teóricos da proposta são discutidos.

Assim, a fim de endereçar esse problema é necessário transladar a AFC do campo meramente teórico para o campo prático. Para tanto, é imperativo padronizar a exposição de certas métricas de avaliação de desempenho na avaliação e/ou comparação de algoritmos, tanto de geração de conceitos quanto de regras. Deve-se levar em consideração a característica da abordagem utilizada, ou seja, se a mesma é sequencial, paralela e/ou distribuída.

Nesse sentido, independente da abordagem de geração de regras empregada, a métrica *tempo de execução* é crucial. De fato, essa é uma das métricas mais importantes, no entanto essa métrica não acrescenta nenhuma informação sem outras métricas que auxiliem a justificar os resultados obtidos. Portanto, para a abordagem sequencial, o tempo de execução será avaliado em conjunto com a dimensionalidade e a densidade do contexto de entrada. Prentende-se com isso justificar o tempo de execução obtido em função desses 2 parâmetros. Já para a abordagem paralela serão avaliadas as mesmas métricas de seu correspondente sequencial. Ademais, será avaliado também a *eficiência* e a *escalabilidade* do algoritmo quando da paralelização do mesmo.

A seguir, na Tabela 3, é apresentado um sumário das métricas para avaliação de desempenho utilizadas neste trabalho.

Tabela 3: Métricas para avaliação de desempenho

Métrica	Expressão	Descrição
tempo sequencial	t_s	tempo de execução sequencial
tempo paralelo	t_p	tempo de execução paralelo
# *threads*	n	nº de *threads* de execução
# núcleos	p	nº de núcleos de computação
speedup	$S(n) = \frac{t_s}{t_p}$	onde n denota o nº de *threads* de execução
eficiência	$E(p) = \frac{S(n)}{p}$	onde $S(n)$ denota o *speedup* alcançado com n *threads*

Nesse ponto, é importante ressaltar que em virtude do alto tempo de execução que se espera, para determinados grupos amostrais, será imposto um limiar para essa métrica. O tempo máximo de execução permitido para os contextos que serão simulados será estipulado em 50 horas. Ultrapassado esse limite, a simulação para o grupo amostral correspondente será interrompida.

4.4 Algoritmo Paralelo

Ambas abordagens estudadas neste trabalho seguem a mesma estrutura sintática, diferindo-se apenas nos pontos de chamada as primitivas de acesso concorrente. Portanto, genericamente, a estrutura do algoritmo pode ser sumarizada nos passos a seguir:

1. Geração do conjunto inicial de atributos
2. Inicialização da lista de regras de implicação
3. Obtenção de regra de implicação
4. Manutenção da lista de regras
5. Geração do conjunto quasi-fechado seguinte

Basicamente, a obtenção da base mínima de implicações consiste na geração de todo o conjunto quasi-fechado de um Contexto Formal. Todavia, como a base mínima é constituída por conjuntos pseudo-fechados, os subconjuntos de interesse são justamente aqueles que não são fechados em relação ao operador de derivação. Como o operador de fecho esta definido em termos do contexto de entrada, pode-se referir aos conjuntos pseudo-fechados ou quasi-fechados como pseudo-intensões ou quasi-intensões, respectivamente.

Os passos 1 e 2 são responsáveis pela inicialização das estruturas de dados empregadas pelo algoritmo, a saber: o subconjunto vazio de atributos (intensão vazia) e

a lista vazia de regras de implicação. Os passos restantes do esboço estão compreendidos dentro de um laço, onde a condição de parada é até que o subconjunto corrente (de quasi-intensões) seja igual ao conjunto completo de atributos.

A cada iteração do algoritmo, um novo subconjunto é gerado e avaliado. Assim, o passo 3 lida com obtenção de prováveis regras. Prováveis, pois como mencionado anteriormente a base mínima de implicações é definida em termos de pseudo-intensões, isto é, quasi-intensões não-fechadas em relação ao operador de derivação. Portanto, na ocasião em que a quasi-intensão for fechada, não resultará em nova regra de implicação.

Dessa forma, caso uma pseudo-intensão seja gerada, a seguir, no passo 4, esse novo subconjunto é adicionado a lista de regras. Novamente, essa estrutura é mantida, em razão da natureza recursiva do algoritmo, uma vez que a base mínima é definida em termos dela mesma, essa estrutura é empregada como retroalimentação no passo seguinte.

Finalmente, no passo 5, a próxima quasi-intensão é gerada. Nesse passo, a lista de regras mantida na etapa anterior, bem como a quasi-intensão corrente são empregadas para geração do próximo subconjunto. Naturalmente, a geração de subconjuntos se dá em ordem lexicográfica, visto que para geração destes, o algoritmo *NextClosure* é utilizado.

Em suma, o conjunto de passos detalhados anteriormente pode ser descrito pelo diagrama a seguir.

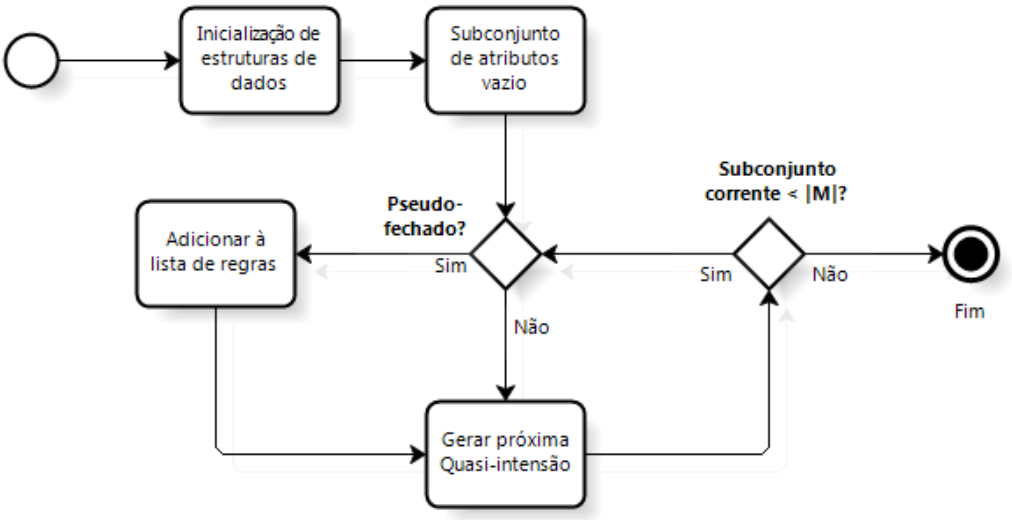


Figura 2: Esquema de funcionamento do algoritmo

4.4.1 Definição de critérios de balanceamento de carga

Uma vez que o presente trabalho lida com a paralelização de um algoritmo, o problema da especificação do tamanho ideal para as cargas de trabalho surge de forma natural. De fato, a paralelização de algoritmos da AFC é um tópico recente e amplamente discutido (FU; NGUIFO, 2004; KENGUE; VALTCHEV; DJAMÉGNI, 2005, 2007; KRAJCA; VYCHODIL, 2009; KRAJCA; OUTRATA; VYCHODIL, 2010), contudo a especificação do tamanho ideal para as partições é um ponto que ainda se encontra em aberto.

Sabe-se que o tamanho empregado para as cargas de trabalho possui impacto relevante sobre o desempenho de algoritmos paralelos/distribuídos. Sendo assim, com a finalidade de minimizar eventuais distorsões que este parâmetro possa infligir sobre os resultados, neste trabalho a especificação do tamanho para as cargas de trabalho leva em conta seu balanceamento. É importante ressaltar que, no presente trabalho, o balanceamento é definido como o tamanho equitativo dos grãos (*i.e.* quantidade de objetos, de atributos e de regras) para cada *thread*.

Como o desempenho esta intimamente relacionado a dimensionalidade, a densidade e ao número de regras gerado, os esforços foram atidos na paralelização dos pontos onde esses parâmetros ocorrem. Portanto, os grãos são definidos em termos da quantidade de objetos, de atributos e de regras. Consequentemente, espera-se que o desempenho observado com a paralelização seja influenciado pela granularidade desses parâmetros.

Apesar do critério de balanceamento selecionado não ser garantidamente o melhor, espera-se que a exposição de métricas e a análise dos resultados experimentais possam auxiliar trabalhos correlatos no cáculdo do tamanho ideal desse parâmetro.

4.4.2 Implementação do algoritmo escalável

Em consonância com o exposto anteriormente, a implementação paralela do algoritmo para obtenção da base mínima de implicações leva em consideração, principalmente, a divisão equitativa das cargas de trabalho. Essa característica foi alcançada com o auxílio das primitivas disponíveis na biblioteca *OpenMP*.

Ressalta-se também que alterações estruturais só foram efetuadas quando estritamente necessário para se alcançar um comportamento paralelo. Dessa forma, a maior parte do esforço de paralelização concentrou-se na identificação e inclusão de diretivas *OpenMP* nos pontos em que a paralelização era possível.

Ademais, conforme pode ser observado nos Algoritmos 1, 2 e 3, a maior parte da lógica desses algoritmos se concentra em operações sobre conjuntos e no uso de estruturas de repetição. Consequentemente, os esforços para implementação da abordagem paralela foram focados em tais estruturas.

Posto isto, para garantir o balanceamento das cargas de trabalho, na paralelização dos laços foi empregada a diretiva `omp for`. Essa diretiva possibilita a execução paralela de estruturas de repetição, dividindo a carga de trabalho entre o time de *threads*, segundo um esquema de escalonamento previamente definido. E, uma vez que pretende-se empregar cargas de trabalho de tamanho equitativo, em conjunto com essa diretiva, foi utilizada como estratégia de escalonamento a cláusula `static`.

Quanto a implementação das estruturas de dados, uma vez que boa parte das operações são concentradas em operações sobre conjuntos, foi empregada uma representação a nível de *bits* para esses conjuntos. Isto foi feito de forma reduzir o espaço necessário para armazenamento de tais estruturas e para maior desempenho nas operações sobre conjuntos.

Finalmente, o conjunto de algoritmos paralelos que compõem a abordagem paralela para obtenção da base canônica de implicações é expressa a seguir nos Algoritmos 4, 5, 6, 7 e 8. Os pontos paralelizados estão compreendidos entre os blocos `Início Região Paralela` e `Fim`. Ressalta-se também que, como a divisão das cargas de trabalho ficam a encargo da biblioteca *OpenMP*, o cálculo das faixas de trabalho específicas de cada *thread* foi omitido nos algoritmos. Portanto, nos pontos onde as variáveis `inicio_faixa` e `fim_faixa` ocorrem, deve ser entendido que estas são calculadas automaticamente, de forma equitativa.

Confome pode ser observado, o Algoritmo 4 apresenta a mesma estrutura de seu correspondente sequencial, pois cada passo desse algoritmo é dependente do passo anterior. Essa característica incorreu em maior dificuldade no processo de paralelização do mesmo. De modo que a paralelização se ateve nas subrotinas `double_prime()` e `lin_closure()`, respectivamente, Algoritmos 5 e 8.

Os algoritmos 6 e 7 constituem as operações de derivação sobre o contexto de entrada, a saber: a extensão e a intensão do conceito. No primeiro algoritmo os grãos são definidos em termos da quantidade de objetos do contexto e do tamanho do subconjunto de atributos $A \subseteq M$. Já para o segundo, os grãos correspondem a quantidade de atributos do contexto e ao tamanho do subconjunto de objetos $B \subseteq G$.

Algoritmo 4: DUQUENNE-GUIGUES base mínima de implicações

Entrada: Contexto Formal $\mathcal{K}(G, M, I)$
Saída: A base canônica $\mathcal{L}$

```
1  L ← ∅
2  A ← ∅
3  enquanto A ≠ M faça
4  |  B ← double_prime(K(G,M,I), A)
5  |  se A ≠ B então
6  |  |  obj ← extent(K(G,M,I), A ∪ B)
7  |  |  supp(i) ← |obj| / |G|
8  |  |  L ← L ∪ {A → B \ A}
9  |  fim
10 |  A ← next_closure(A, lin_closure )
11 fim
12 retorna L
```

Algoritmo 5: DOUBLE PRIME operador de derivação

Entrada: Contexto Formal $\mathcal{K}(G, M, I)$, subconjunto $A \subseteq M$
Saída: fecho de A

```
1  B ← extent(K(G,M,I), A)
2  C ← intent(K(G,M,I), B)
3  retorna C
```

Na linha 7 desses dois algoritmos, a variável `tid` corresponde ao identificador da *thread* em execução. Mais especificamente, o pseudo-código compreendido entre os blocos `Início Região Paralela` e `Fim`, será executado em paralelo por um time de *threads* de execução. Cada *thread* pertencente ao time recebe um identificador único, de forma a viabilizar a união dos resultados parciais na linha 21.

Novamente, o cálculo das faixas de trabalho específicas de cada *thread* de execução é definido de forma automática e é calculado de acordo com os limites de domínio do laço. No caso do Algoritmo 6, faixas no intervalo $0 \leq i < |G|$ são calculadas. Para o Algoritmo 7, faixas entre $0 \leq i < |M|$.

O Algoritmo 8, por sua vez, corresponde a versão paralela do operador de fecho de conjuntos quasi-fechados. Nesse algoritmo, os grãos do primeiro bloco paralelo, compreendido entre as linhas 1 e 11, são definidos em termos do número de atributos do contexto e da quantidade de regras (*i.e.* obtidas até o momento). Nesse bloco, as faixas de trabalho do time de *threads* são definidas dentro do intervalo $1 \leq x \leq |M|$. Já para o segundo bloco paralelo (linhas $18-31$), a granularidade é definida pelo tamanho dos subconjuntos `T` e `new_closure`.

Algoritmo 6: EXTENT extensão do conceito

Entrada: Contexto Formal $\mathcal{K}(G, M, I)$, subconjunto $A \subseteq M$
Saída: Conjunto de objetos que possuem todos os elementos de A

```
1  NUM_THREADS ← num_threads()
2  A' ← {0, 0, ..., 0_{G-2}, 0_{G-1}}
3  para i ← 0 até NUM_THREADS faça
4  |  objetos[i] ← {0, 0, ..., 0_{G-2}, 0_{G-1}}
5  fim
6  Início Região Paralela
7  |  tid ← thread_id()
   |  /* Faixas no intervalo 0 ≤ i < |G|.                                  */
8  |  para i ← inicio_faixa até fim_faixa faça
9  |  |  todos ← V
10 |  |  para cada j ∈ A faça
11 |  |  |  se (i, j) ∉ I então
12 |  |  |  |  todos ← F
13 |  |  |  fim
14 |  |  fim
15 |  |  se todos = V então
16 |  |  |  objetos[tid] ← objetos[tid] ∪ {i}
17 |  |  fim
18 |  fim
   |  /* Ponto de sincronização:  apenas a thread com identificador igual
   |     a 0 (zero) executa este trecho.                                   */
19 |  se tid = 0 então
20 |  |  para i ← 0 até NUM_THREADS faça
21 |  |  |  A' ← A' ∪ objetos[tid]
22 |  |  fim
23 |  fim
24 Fim
25 retorna A'
```

Diferente dos demais pontos paralelizados, a segunda região paralela apresenta paralelismo de seção. Na implementação *OpenMP* esse comportamento é descrito pelo uso de diretivas `omp section`, que correspondem a um tipo não-iterativo de paralelismo, onde cada seção paralela é atribuída a uma *thread* de execução[1]. Esse tipo de paralelismo foi adotado para esse bloco em virtude da baixa granularidade apresentada nessa região.

Por fim, apesar de ser possível paralelizar o laço da linha 35, o mesmo não foi paralelizado, em virtude de sua baixa granularidade.

[1]De acordo com a documentação *OpenMP*, pode ocorrer de uma mesma *thread* executar mais de uma seção paralela, se a implementação desta permitir.

Algoritmo 7: INTENT intensão do conceito

Entrada: Contexto Formal $\mathcal{K}(G, M, I)$, subconjunto $B \subseteq G$
Saída: Conjunto de atributos que possuem todos os elementos de B

```
1  NUM_THREADS ← num_threads()
2  B' ← {0, 0, ..., 0_{M-2}, 0_{M-1}}
3  para i ← 0 até NUM_THREADS faça
4  |  atributos[i] ← {0, 0, ..., 0_{M-2}, 0_{M-1}}
5  fim
6  Início Região Paralela
7  |  tid ← thread_id()
   |  /* Faixas no intervalo 0 ≤ i < |M|.                              */
8  |  para i ← inicio_faixa até fim_faixa faça
9  |  |  todos ← V
10 |  |  para cada j ∈ B faça
11 |  |  |  se (j, i) ∉ I então
12 |  |  |  |  todos ← F
13 |  |  |  fim
14 |  |  fim
15 |  |  se todos = V então
16 |  |  |  atributos[tid] ← atributos[tid] ∪ {i}
17 |  |  fim
18 |  fim
   |  /* Ponto de sincronização:  apenas a thread com identificador igual
   |     a 0 (zero) executa este trecho.                                */
19 |  se tid = 0 então
20 |  |  para i ← 0 até NUM_THREADS faça
21 |  |  |  B' ← B' ∪ atributos[tid]
22 |  |  fim
23 |  fim
24 Fim
25 retorna B'
```

Algoritmo 8: LinClosure

Entrada: Uma lista $\mathcal{L} \leftarrow [\mathcal{L}[1], \ldots, \mathcal{L}[n]]$ de implicações em M e um conjunto $X \subseteq M$
Saída: Fecho $\mathcal{L}(X)$ de X

```
1  Início Região Paralela
   |  /* Faixas no intervalo 0 ≤ x < |M|.                                  */
2  |  para x ← inicio_faixa até fim_faixa faça
3  |  |  avoid[x] ← {1, ..., n}
4  |  |  para i ← 1 até n faça
5  |  |  |  A → B ← L[i]
6  |  |  |  se x ∈ A então
7  |  |  |  |  avoid[x] ← avoid[x] \ {i}
8  |  |  |  fim
9  |  |  fim
10 |  fim
11 Fim
12 used_imps ← ∅
13 old_closure ← {−1}
14 new_closure ← X
15 enquanto new_closure ≠ old_closure faça
16 |  old_closure ← new_closure
17 |  T ← M\new_closure
18 |  Início Região Paralela
19 |  |  Seção Paralela
20 |  |  |  tmp1 ← {1, 1, ..., 1_{M−1}, 1_M}
21 |  |  |  para cada x ∈ T faça
22 |  |  |  |  tmp1 ← tmp1 ∩ avoid[x]
23 |  |  |  fim
24 |  |  Fim
25 |  |  Seção Paralela
26 |  |  |  tmp2 ← {0, 0, ..., 0_{M−1}, 0_M}
27 |  |  |  para cada x ∈ new_closure faça
28 |  |  |  |  tmp2 ← tmp2 ∪ avoid[x]
29 |  |  |  fim
30 |  |  Fim
31 |  Fim
32 |  usable_imps ← tmp1 ∩ tmp2
33 |  use_now_imps ← usable_imps \ used_imps
34 |  used_imps ← usable_imps
35 |  para todo i ∈ use_now_imps com A → B ← L[i] faça
36 |  |  new_closure ← new_closure ∪ B
37 |  fim
38 fim
39 L(x) ← new_closure
40 retorna L(x)
```

5 ANÁLISE DOS RESULTADOS

Este capítulo apresenta os resultados dos experimentos conduzidos para o conjunto de contextos formais considerados.

5.1 Resultados Experimentais

Inicialmente, ressalta-se que a implementação do algoritmo para obtenção da base mínima de implicações utilizada neste trabalho foi compilada no próprio ambiente onde as simulações seriam realizadas (vide Tabela 4). Isto foi feito de modo a minimizar distorsões relacionadas a versionamento de compilador e de suas bibliotecas. Posto isto, a seguir, na Tabela 5, é apresentado o desempenho da abordagem sequencial. Para cada contexto sintético são apresentados os tempos de execução obtidos com a variação da densidade; hífens denotam que aquela simulação foi abortada em virtude do limiar definido na metodologia. Alguns contextos que ultrapassaram o limiar são mostrados, e são indicados com asteriscos.

Tabela 4: Configuração do ambiente experimental

Item	Valor
Processador	Intel Xeon E5430 2.6GHz / 8 núcleos
Memória *Cache*	6144KB
Memória Primária	8GB
Sistema Operacional	RHEL v4.1.2-44
Compilador	GCC v4.1.2
Implementação *OpenMP*	v2.5

Repare que, de modo geral, para maior parte dos casos, a obtenção da base mínima apresenta pior caso quando a densidade do contexto de entrada é de 50% (KUZNETSOV; OBIEDKOV, 2008), o que é notadamente divergente do comportamento observado na obtenção de conceitos formais, onde o pior caso é verificado quando a entrada apresenta densidade máxima. Em especial, para o contexto de 15 atributos e 5000 objetos esse comportamento não é verificado. Em parte, por se tratar de um contexto relativamente pequeno, mas principalmente por este necessitar de uma quantidade de iterações (*i.e.* quasi-intensões) muito próxima dos contextos com densidades de 70% e 70,12% (vide Tabela 2).

Outro ponto a ser observado, diz respeito aos contextos 15×5000 com densidades

Tabela 5: Média do tempo sequencial (em segundos) para construção da base canônica dos contextos artificiais da Tabela 2

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	tempo (em segundos)				
15×1000	1,35	3,81	12,50	9,33	9,31
15×5000	27,35	24,25	41,58	43,40	43,39
15×10000	69,01	-	82,56	-	85,57
20×1000	4,07	122,22	2350,07	1344,65	408,35
20×5000	49,41	836,65	7360,39	1851,14	1897,59
20×10000	226,77	1130,12	9382,43	3677,29	3711,39
25×1000	8,40	2149,90	671753,12*	-	-
25×5000	211,48	54957,18	-	-	-
25×10000	254,07	165507,72	-	-	-

mínima e de 30%. Nesses dois casos, percebe-se que o contexto com densidade mínima (*i.e.* 29,88%) apresentou tempo de execução superior ao seu correspondente com 30%. Isto é justificado pela proximidade das densidades. Além é claro, da entrada com densidade mínima ter proporcionado uma quantidade superior tanto de quasi-intensões quanto de regras, o que sugere que, assim como no trato de conceitos formais, a relação de incidência do contexto tem peso relevante na obtenção da base mínima de implicações.

Ainda na Tabela 5, nota-se que para os contextos com mais de 20 atributos é nítida a explosão no tempo de execução decorrente do aumento da densidade da entrada. Também é possível observar uma certa tendência em relação ao tempo de execução obtido mediante a variação da densidade do contexto. De certo, pode-se afirmar que, após uma taxa de preenchimento de 50%, há um declíneo no tempo de execução. A Tabela 6 ajuda a interpretar tal comportamento.

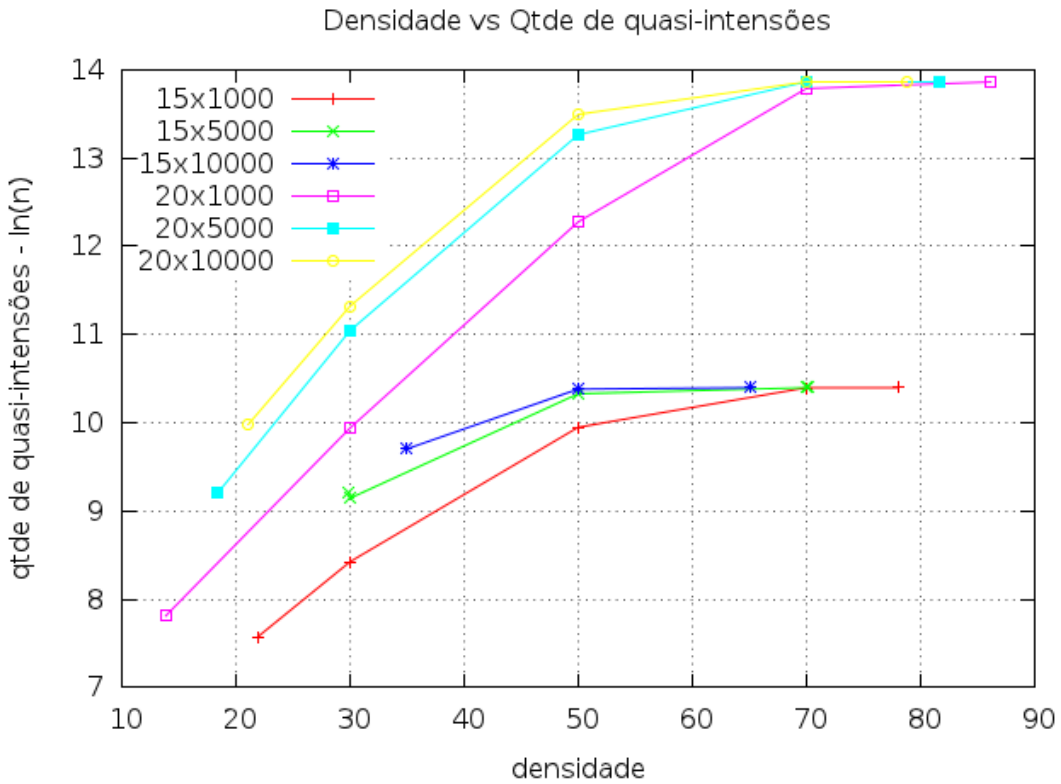
Tabela 6: Média do nº de regras obtido para cada grupo amostral

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	nº de regras (n)				
15×1000	952	1515	1389	7	0
15×5000	4948	4324	594	0	0
15×10000	6353	-	208	-	0
20×1000	1466	7830	22820	3972	0
20×5000	4953	20614	33172	53	0
20×10000	11702	19693	31385	5	0
25×1000	1724	28074	-	-	-
25×5000	10287	94810	-	-	-
25×10000	8848	137028	-	-	-

Comparando os dados da Tabela 6 com os tempos de execução apresentados na Tabela 5, percebe-se que o desempenho do algoritmo esta intimamente relacionado com o número de regras obtido. De modo geral, a medida que a densidade aumenta, isto é, da mínima possível até 50%, o número de regras também cresce. Em contrapartida, após 50%, há um declíneo do número de regras. Em especial, quando empregada a densidade máxima, para todos os contextos utilizados, o número de regras foi sempre nulo. Isto ocorre, pois nesse caso todos os subconjuntos gerados são fechados em relação ao operador de derivação. Todavia, os subconjuntos de interesse, para a construção da base canônica de implicações, são justamente aqueles que não são fechados (vide Algoritmo 3).

Apesar do número de regras afetar o desempenho do algoritmo, este fator não confere um maior tempo de execução. Um exemplo dessa situação pode ser observado no contexto 15×5000 com densidade mínima. Nesse caso, o contexto com essa configuração foi o que apresentou maior número de regras, no entanto esta característica não conferiu o maior tempo gasto. Isto mostra que embora o desempenho também esteja sujeito ao número de regras, essa medida não contribui isoladamente com o incremento do tempo de execução. De fato, tal como pode ser observado no gráfico expresso na Figura 3, a obtenção do conjunto de regras também esta condicionada à quantidade de subconjuntos de quasi-intensões obtidas do contexto de entrada.

Figura 3: Variação da densidade do contexto e o nº de quasi-intensões correspondente



No gráfico da Figura 3 é fácil notar que o número de subconjuntos (*i.e.* quasi-intensões) cresce com o incremento da densidade do contexto. De fato, esse número é exponencial na quantidade de atributos da entrada, onde no caso em que o contexto apresenta densidade máxima, o número de quasi-intensões correspondente atinge seu maior valor (*i.e.* $2^{|M|}$). Outra consideração importante, se refere a relação entre o número de subconjuntos gerados e o de objetos. Tal como os resultados mostram, a quantidade de objetos tem impacto linear no número de quasi-intensões geradas.

Uma análise mais atenta aos resultados da Tabela 6 e do gráfico da Figura 3 revela ainda, que o aumento do número de objetos no contexto de entrada torna o declíneo do número de regras mais acentuado, a medida que a densidade se aproxima do valor máximo. Isto é esperado, já que o aumento do número de objetos torna mais difícil o processo de obtenção de regras, seja pelo incremento no número de subconjuntos avaliados ou mesmo pelo esforço demandado para se realizar operações de derivação.

A Tabela 7 apresenta o desvio-padrão do tempo de execução dos grupos amostrais empregados nas simulações. Nessa Tabela, os resultados explicitam o impacto da relação de incidência do contexto sobre o desempenho do algoritmo. Além disso, mostra que apesar de contextos pertencentes a um mesmo grupo amostral serem similares em dimensão e densidade, ainda assim se referem a contextos diferentes.

Tabela 7: Desvio-padrão do tempo de execução dos grupos amostrais na abordagem sequencial

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	tempo (em segundos)				
15×1000	0,02	0,13	0,29	0,19	0,02
15×5000	0,01	0,12	0,15	0,13	0,13
15×10000	0,29	-	0,06	-	0,05
20×1000	0,08	2,50	40,44	184,27	1,20
20×5000	0,14	7,90	227,53	7,06	1,51
20×10000	0,32	19,24	589,28	4,72	9,58
25×1000	0,10	51,52	-	-	-
25×5000	0,78	515,57	-	-	-
25×10000	0,92	15013,92	-	-	-

Na Tabela 7, observa-se que o incremento no desvio-padrão esta intimamente relacionado ao aumento do número de atributos da entrada. Isto é nítido para os contextos com mais de 20 atributos. Ademais, com auxílio da Tabela 8, que apresenta o grau de dispersão da quantidade de regras de cada grupo amostral, é possível perceber que o tempo de execução também esta relacionado com o número de regras gerado, corroborando desta

forma com as observações levantadas acerca da densidade da entrada.

Tabela 8: Desvio-padrão do nº de regras obtido para cada grupo amostral

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	nº de regras (n)				
15×1000	2	26	69	2	0
15×5000	0	24	65	0	0
15×10000	4	-	33	-	0
20×1000	7	110	479	903	0
20×5000	30	317	1784	33	0
20×10000	11	60	3123	3	0
25×1000	18	462	-	-	-
25×5000	6	136	-	-	-
25×10000	47	1985	-	-	-

As tabelas 9 e 10 apresentam o *speedup* e a eficiência dos algoritmos 5 e 8 quando empregados 8 núcleos de computação. Como o esforço de paralelização foi focado nessas duas rotinas, foi possível mensurar a contribuição desses trechos no desempenho final da proposta.

Tabela 9: *Speedup* e Eficiência alcançados em DoublePrime com 8 *threads* de execução

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	Ganho / Eficiência				
15×1000	0,901 / 0,113	1,079 / 0,135	1,389 / 0,174	1,522 / 0,19	1,557 / 0,195
15×5000	1,128 / 0,141	1,164 / 0,146	1,599 / 0,2	1,842 / 0,23	1,836 / 0,229
15×10000	1,229 / 0,154	-	1,491 / 0,186	-	1,835 / 0,229
20×1000	0,932 / 0,116	1,333 / 0,167	1,589 / 0,199	1,776 / 0,222	1,644 / 0,205
20×5000	1,209 / 0,151	1,554 / 0,194	1,858 / 0,232	1,813 / 0,227	2,2 / 0,275
20×10000	1,297 / 0,162	1,626 / 0,203	1,889 / 0,236	1,802 / 0,227	1,66 / 0,208
25×1000	1,014 / 0,127	1,554 / 0,194	1,836 / 0,23	-	-
25×5000	1,293 / 0,162	1,825 / 0,228	-	-	-
25×10000	1,432 / 0,179	1,894 / 0,237	-	-	-

Percebe-se na Tabela 9 que o *speedup* de DoublePrime esta relacionado ao número de quasi-intensões do contexto de entrada. Nota-se também que o aumento da densidade proporciona melhora no desempenho dessa rotina. Na Tabela 10, por outro lado, observa-se que quanto maior o número de pseudo-intensões obtidas, maior o *speedup* de LinClosure.

Na Tabela 11 são apresentados os tempos obtidos com a paralelização da base mínima de implicações. São mostrados para cada contexto sintético, os tempos de execução obtidos com a variação da densidade, bem como do número de *threads*. Também nessa Tabela, hífens denotam que o contexto de entrada com aquela densidade e número

Tabela 10: *Speedup* e Eficiência alcançados em LinClosure com 8 *threads* de execução

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	Ganho / Eficiência				
15 × 1000	1,805 / 0,226	1,907 / 0,238	2,519 / 0,315	0,096 / 0,012	0,075 / 0,009
15 × 5000	3,426 / 0,428	3,152 / 0,394	1,315 / 0,164	0,073 / 0,009	0,079 / 0,01
15 × 10000	3,667 / 0,458	-	0,414 / 0,052	-	0,08 / 0,01
20 × 1000	2,441 / 0,305	3,951 / 0,494	4,66 / 0,582	4,186 / 0,523	0,088 / 0,011
20 × 5000	3,427 / 0,428	4,423 / 0,553	5,007 / 0,626	0,236 / 0,029	0,087 / 0,011
20 × 10000	3,929 / 0,491	4,373 / 0,547	5,115 / 0,639	0,093 / 0,012	0,1 / 0,013
25 × 1000	2,744 / 0,343	4,398 / 0,55	3,662 / 0,458	-	-
25 × 5000	3,774 / 0,472	4,03 / 0,504	-	-	-
25 × 10000	3,855 / 0,482	3,845 / 0,481	-	-	-

de *threads*, não foi simulado em virtude do tempo de execução ter extrapolado o limiar definido na metodologia (vide Seção 4.3).

Tabela 11: Média do tempo paralelo (em segundos) para construção da base canônica dos contextos artificiais da Tabela 2

(continua)

Contexto	# threads	Densidade (%)				
		mín.	30	50	70	máx.
		tempo (em segundos)				
15 × 1000	2	1,33	3,30	10,86	9,79	9,84
15 × 5000		24,32	22,28	40,45	44,92	46,36
15 × 10000		66,28	-	86,01	-	77,81
20 × 1000		3,25	73,90	1313,53	888,01	403,27
20 × 5000		38,25	508,71	4306,58	1660,34	1599,46
20 × 10000		171,06	745,97	5954,80	3228,70	3409,13
25 × 1000		6,24	1241,77	-	-	-
25 × 5000		144,78	35016,49	-	-	-
25 × 10000		180,25	101669,18	-	-	-
15 × 1000	4	1,43	2,43	7,75	7,03	7,11
15 × 5000		16,68	15,47	28,49	29,56	30,53
15 × 10000		43,80	-	56,76	-	56,95
20 × 1000		2,32	45,82	756,00	554,59	261,94
20 × 5000		24,71	307,21	2473,64	1048,74	1025,64
20 × 10000		107,71	454,05	3450,38	2159,62	2322,41
25 × 1000		4,32	749,47	-	-	-
25 × 5000		90,68	24259,85	-	-	-

(conclusão)

Contexto	# threads	Densidade (%)				
		mín.	30	50	70	máx.
		tempo (em segundos)				
25 × 10000	4	115,76	80372,78	-	-	-
15 × 1000	8	0,94	2,13	6,82	7,33	7,76
15 × 5000		15,19	13,81	26,60	24,94	25,16
15 × 10000		41,36	-	54,95	-	50,05
20 × 1000		2,60	35,01	537,62	433,82	297,05
20 × 5000		21,39	235,88	1805,16	1066,64	913,23
20 × 10000		90,57	364,94	2656,57	2087,76	2277,62
25 × 1000		3,51	505,03	183897,60*	-	-
25 × 5000		68,14	13872,32	-	71873,45	18403,50
25 × 10000		90,78	43701,65	-	-	-
15 × 1000	16	4,17	8,30	25,92	33,18	33,48
15 × 5000		30,15	27,70	58,52	58,39	57,06
15 × 10000		68,54	-	101,92	-	91,57
20 × 1000		8,08	69,32	771,35	1180,88	1087,44
20 × 5000		42,44	340,78	2456,46	1981,20	2048,81
20 × 10000		141,62	523,04	3611,47	3393,92	3385,14
25 × 1000		12,16	647,78	-	-	-
25 × 5000		110,00	14939,25	-	-	-
25 × 10000		146,39	45030,98	-	-	-

Da Tabela 11, nota-se que assim como seu correspondente sequencial, o tempo paralelo também depende da densidade do contexto de entrada. Uma comparação dos tempos paralelos com os sequenciais, expressos na Tabela 5, revela ainda que o desempenho só é significativo para os contextos com 15 atributos quando empregadas 4 ou mais *threads* de execução.

De fato, não só o desempenho da implementação sequencial, mas também da versão paralela dependem da densidade do contexto. Assim, nas simulações realizadas observa-se que o *speedup* máximo[1] é obtido quando a matriz de incidência do contexto considerado

[1] *Speedup* máximo em comparação com os demais contextos de mesma dimensão, mas com densidades diferentes.

apresenta densidade esparsa e número de regras relativamente alto, conforme ilustrado pelos dados da Tabela 12, que apresenta o ganho obtido com 8 *threads* de execução.

Tabela 12: *Speedup* alcançado com 8 *threads* e 8 núcleos disponíveis

Contexto	Densidade (%)				
	mín.	30	50	70	máx.
	Ganho				
15×1000	1,439	1,785	1,832	1,273	1,20
15×5000	1,801	1,756	1,563	1,740	1,724
15×10000	1,669	-	1,502	-	1,710
20×1000	1,565	3,491	4,371	3,10	1,375
20×5000	2,311	3,547	4,077	1,735	2,078
20×10000	2,504	3,097	3,532	1,761	1,630
25×1000	2,397	4,257	3,653	-	-
25×5000	3,103	3,962	-	-	-
25×10000	2,799	3,787	-	-	-

Na Tabela 12, tomando-se a constatação acerca da obtenção do desempenho máximo do algoritmo, é possível notar que o *speedup* atinge seu pico quando o contexto de entrada apresenta elevado número de regras e densidade esparsa. Isto é corroborado pelos contextos 15×5000 com densidade mínima, 25×1000 e 25×5000 com densidade 30% e 15×1000, 20×1000, 20×5000 e 20×10000 com densidade 50%. Nesses casos, tal como mostrado na Tabela 6, os contextos considerados apresentaram elevada quantidade de regras, em comparação com seus correspondentes nas demais densidades.

Reforçando essas constatacões, percebe-se também que os contextos que apresentaram resultados[2] com 50% de densidade (*i.e.* 15×5000 e 15×10000), mas que resultaram em menor quantidade de regras obtiveram menor *speedup*. Isto esta de acordo com a alegação anterior acerca do desempenho da abordagem paralela. De fato, como o tamanho dos grãos na abordagem paralela são definidos em termos da quantidade de objetos, de atributos e de regras, o baixo indíce de regras nesse caso incorreu em grãos mais finos e, portanto, menor desempenho.

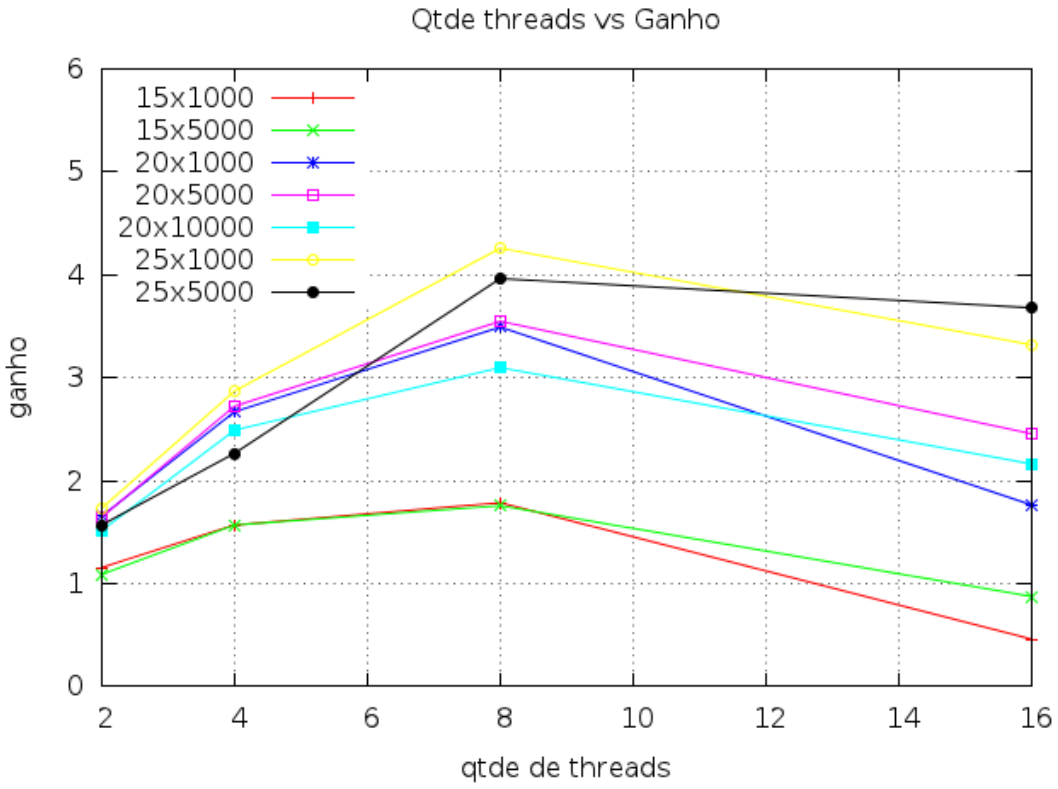
Outro ponto que chama a atenção é o resultado obtido para o contexto 15×10000, com densidade mínima. Diferente dos demais contextos esparsos que produziram muitas regras, esse contexto, com 8 *threads*, apresentou menor *speedup* do que o contexto com densidade máxima, que gerou menos regras. Contudo, esse comportamento se deve ao número de *threads* utilizadas nesta simulação, pois no cenário em que foram empregadas 4 *threads* de execução, tal contexto de entrada apresentou maior desempenho do que os

[2]Contextos sintéticos que apresentaram tempo de execução inferior ao limiar definido.

demais. Mais especificamente, nessa situação, as simulações com contextos 15×10000, com densidades mínima, 50% e máxima apresentaram, respectivamente, *speedup* de $1,575$, $1,455$ e $1,502$.

Os gráficos das Figuras 4, 5 e 6 mostram o desempenho da abordagem paralela com o incremento do número de *threads* de execução para os contextos sintéticos com 30%, 50% e 70% de densidade, respectivamente. Os gráficos explicitam a escalabilidade alcançada com a paralelização da base mínima de implicações na obtenção do conjunto de regras. São mostrados apenas os contextos que apresentaram tempo de execução menor ou igual ao limiar definido na metodologia.

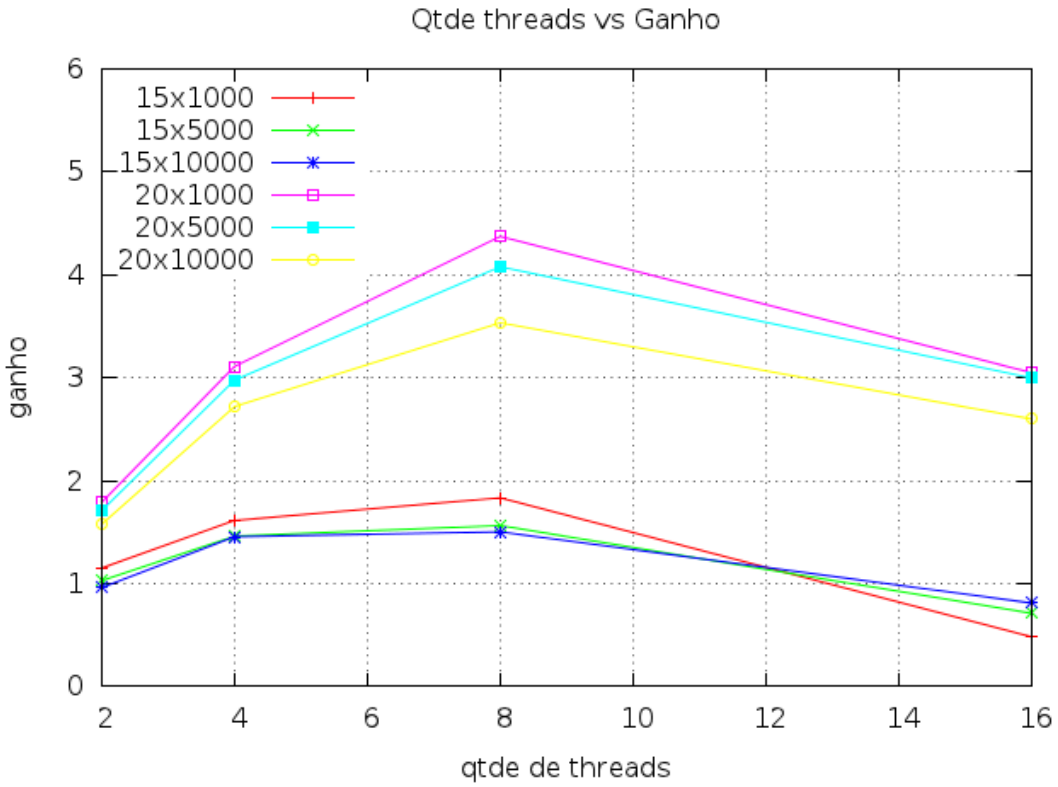
Figura 4: *Speedup* obtido com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 30%



Na Figura 4, observa-se que o *speedup* máximo é obtido quando o número de *threads* é igual a 8. Nota-se também, que o desempenho obtido esta intimamente relacionado ao tamanho dos grãos. Mais especificamente, contextos com maior quantidade de atributos e que geraram mais regras, obtiveram maior *speedup*.

Contudo, nota-se que há perda de desempenho decorrente do aumento do número de objetos, para os contextos que apresentaram elevada quantidade de regras. Esse comportamento pode ser claramente observado nos contextos com 15, 20 e 25 atributos, quando empregadas até 8 *threads* de execução. Isto se deve ao cálculo da extensão na

Figura 5: *Speedup* obtido com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 50%



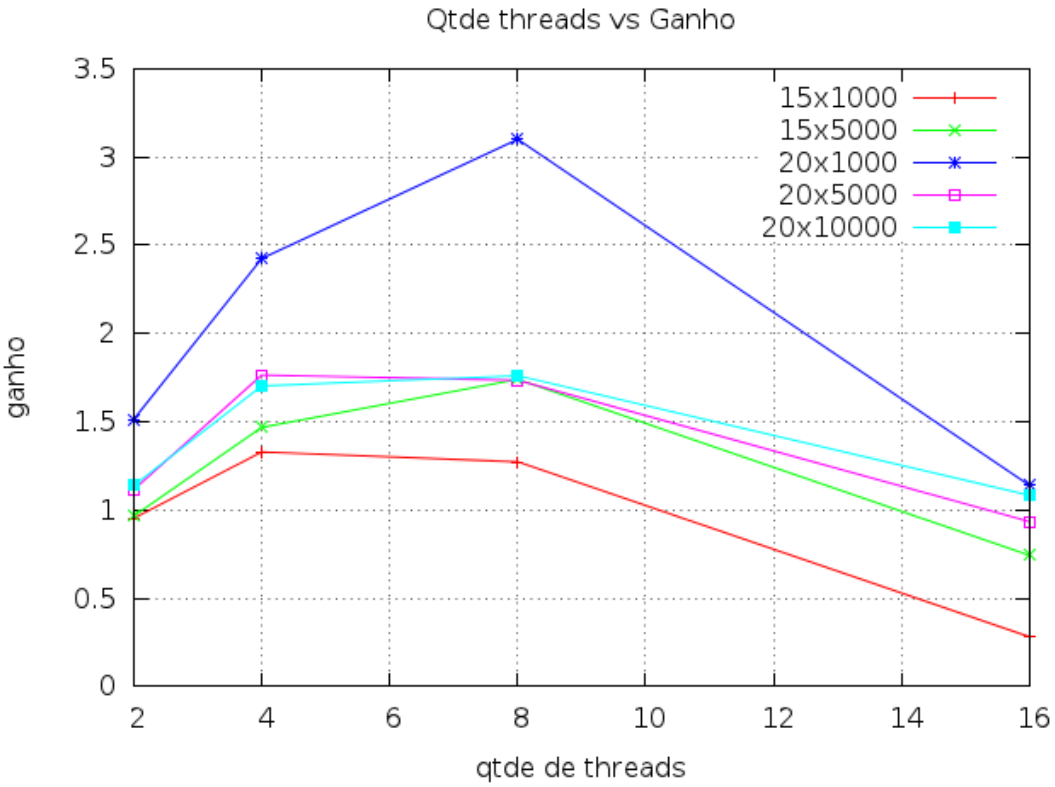
linha 7 do Algoritmo 4, onde é calculado o suporte (*i.e.* número de objetos envolvidos na implicação) da regra obtida. Como o cálculo da extensão lida com objetos, quanto maior este parâmetro, maior o esforço envolvido em tal operação.

O declíneo de desempenho decorrente do aumento do número de objetos, também pode ser visualizado no gráfico da Figura 5, que apresenta a escalabilidade da abordagem paralela para os contextos com 50% de densidade. Esse comportamento é apresentado apenas para os contextos com 20 atributos, já que, como mencionado anteriormente, os contextos com 15 atributos apresentaram redução do número de regras com o aumento da quantidade de objetos (vide Tabela 6).

A Figura 6, por sua vez, mostra a escalabilidade para os contextos com 70% de densidade. Nesse gráfico é nítido a relação entre o *speedup* e o número de regras, onde o contexto 20×1000, por apresentar número de regras superior aos demais contextos, obteve o maior desempenho.

No gráfico da Figura 6, também se nota que o *speedup* máximo é alcançado com 4 e 8 *threads* de execução. Na situação em que foram empregadas 4 *threads*, os contextos 15×1000 e 20×5000 obtiveram melhores resultados. Por outro lado, quando empregadas

Figura 6: *Speedup* obtido com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 70%

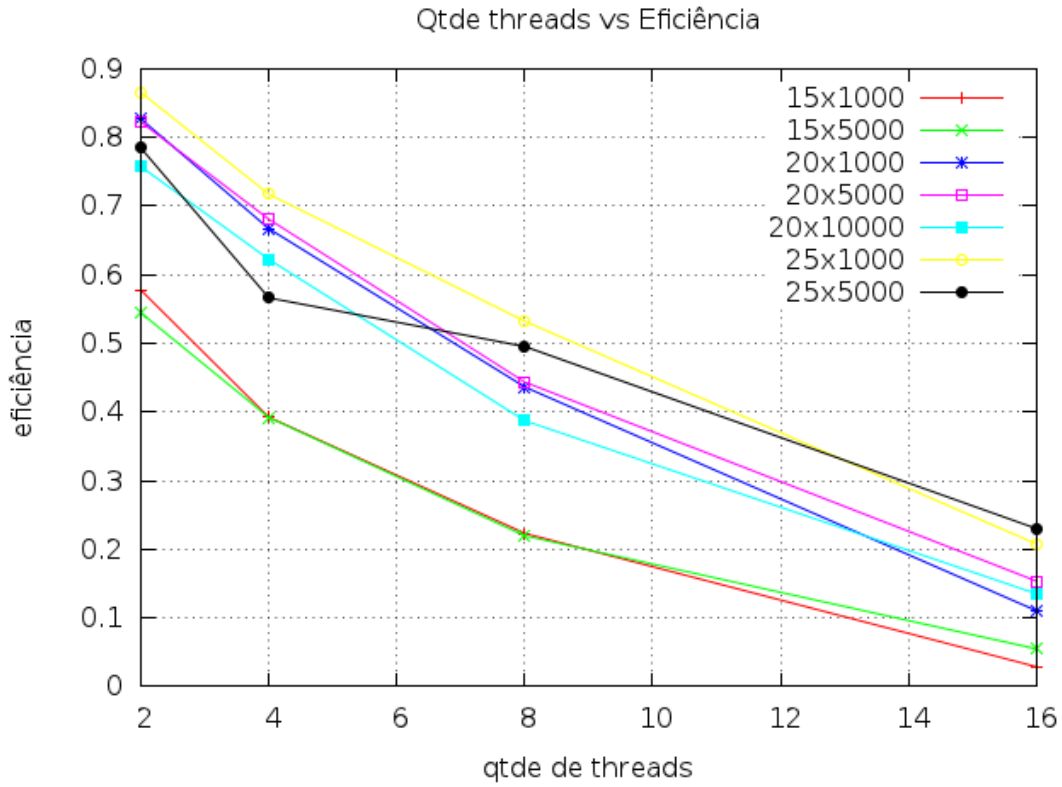


8 *threads*, esses contextos apresentaram ligeiro declíneo no *speedup*, enquanto os demais contextos mostraram aumento no desempenho. Também esse comportamento é justificado pelo tamanho dos grãos. Como nesse cenário poucas regras foram geradas, o desempenho da abordagem paralela ficou a cargo do paralelismo alcançado no Algoritmo 5.

Também é importante ressaltar o comportamento da abordagem paralela quando empregadas 16 *threads* de execução, conforme pode visto nos gráficos das Figuras 4, 5 e 6. Em tal situação, para todas as densidades consideradas, houve redução no ganho obtido. Esse comportamento é justificado, pois esse cenário incorreu em maior competição entre as *threads*, uma vez que o número de núcleos disponíveis era de 8 (vide Tabela 4). Percebe-se também que alguns contextos, nesse cenário, apresentaram piora, em relação a abordagem sequencial, com a paralelização (*i.e.* 15×1000 e 15×5000 com 30%; 15×1000, 15×5000 e 15×10000 com 50%; 15×1000, 15×5000 e 20×5000 com 70%), o que esta relacionado com a granularidade.

A seguir, nas Figuras 7, 8 e 9, a eficiência adquirida com o aumento do número de *threads* é apresentada, para os contextos com 30%, 50% e 70% de densidade, respectivamente.

Figura 7: Eficiência obtida com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 30%



Tal como os resultados mostram, nas Figuras 7, 8 e 9, é nítido o declíneo da eficiência em função do incremento do número de *threads*, quando mantido o tamanho da entrada. Além disso, nota-se que o aumento do número de atributos no contexto de entrada, proporciona uma melhora para essa métrica.

Figura 8: Eficiência obtida com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 50%

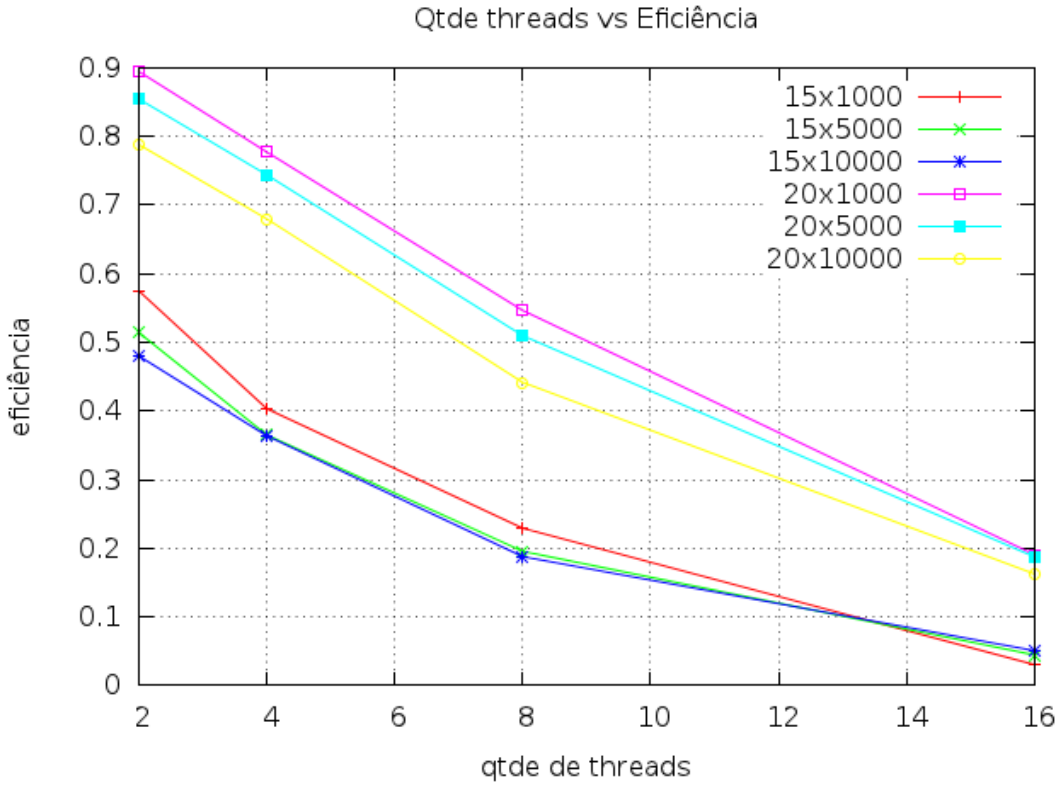
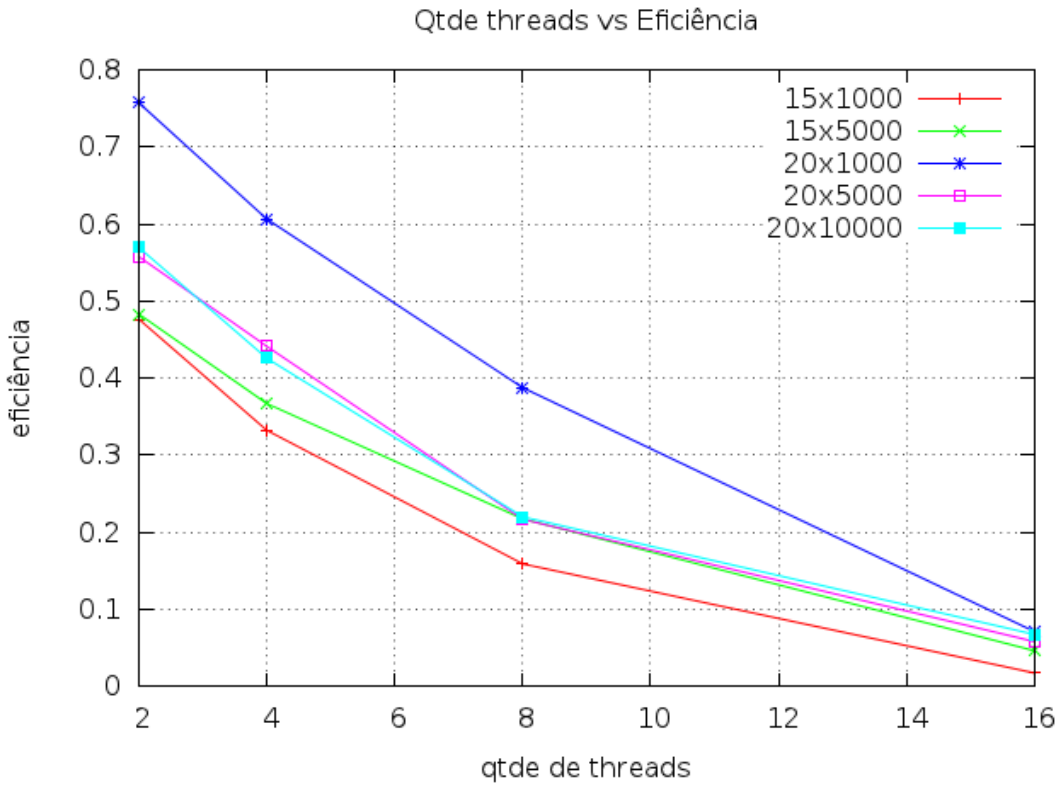


Figura 9: Eficiência obtida com o aumento do nº de *threads* de execução para os contextos com taxa de preenchimento de 70%



5.2 Desafio de Dresden

Conforme mencionado anteriormente, há especial interesse dos pesquisadores da área no desenvolvimento de técnicas e de algoritmos capazes de manipular contextos densos e de alta dimensionalidade. Este fato deu origem ao recorrente desafio, exposto na 4ª edição da *ICFCA*, que menciona as dificuldades decorrentes da manipulação de contextos com tais características. Para exemplificar o problema, naquela ocasião foi mencionado um contexto da ordem de 120000 objetos por 70000 atributos.

Porém, neste trabalho, tal como demonstrado nos experimentos, o algoritmo para obtenção da base mínima de implicações tem pior caso no número de atributos do contexto. Assim, nesta seção são realizados experimentos com foco na alta dimensionalidade no número de objetos. Os experimentos foram conduzidos em contextos de 25 atributos com densidades mínima e 30%. O objetivo dessas simulações é verificar se a abordagem paralela consegue lidar com os contextos especificados, segundo o limiar definido na metodologia.

A seguir, na Tabela 13, são apresentados os tempos de execução e o número de regras para os contextos formais considerados.

Tabela 13: Simulações com 8 *threads* de execução

$M \times G$	Densidade (%)	tempo (em segundos)	nº de regras
25×30000	17,57	1131,40	38496
25×30000	30	174873,88	227064
25×50000	18,54	1466,93	32285
25×50000	30	372129,97*	354608
25×70000	19,05	28669,74	175505
25×70000	30	-	
25×90000	20,15	26565,92	155505
25×90000	30	-	
25×120000	21,11	22704,28	125603
25×120000	30	-	

Tal como discutido na Seção 5.1, os resultados da Tabela 13 explicitam o impacto da densidade do contexto de entrada sobre o desempenho do algoritmo para obtenção da base mínima de implicações. Contudo, não só esse parâmetro impacta no tempo de execução. Também o parâmetro número de regras possui influência relevante sobre essa métrica. Isto porque para obtenção do conjunto de implicações, as regras já obtidas são empregadas para a geração das demais (vide Algoritmo 2).

Analisando os resultados apresentados, é fácil verificar o comportamento do algoritmo em relação aos parâmetros densidade e número de regras. Por exemplo, nos resultados dos contextos com densidade mínima, percebe-se um aumento significativo no

tempo de execução a partir de 70000 objetos. Esse aumento se deve, principalmente, ao incremento do número de regras. De fato, dos contextos com densidade mínima considerados nessa simulação, aquele que apresentou maior tempo de execução foi justamente o que gerou maior número de regras (*i.e.* 25×70000).

Tais observações também são válidas para os demais contextos com 30% de densidade. Nesse caso, observa-se que o contexto que apresentou maior tempo de execução, também foi o que propiciou maior número de regras (*i.e.* 25×50000). Porém, infelizmente, em razão do limiar definido, não foi possível realizar testes com um número maior de objetos para essa densidade.

6 ESTUDO DE CASO

Existe um crescente interesse da comunidade científica no estudo de rede sociais. Essas redes podem subsidiar uma miríade de aplicações, entre as quais a detecção e previsão de eventos, a caracterização do comportamento de usuários, a identificação de comunidades, a implementação de sistemas de recomendação mais eficazes, etc.

No campo da Análise Formal de Conceitos, as redes sociais apresentam oportunidades, mas também desafios, dependendo da forma como o problema é modelado (ROME; HARALICK, 2005). Isto porque, como já se sabe, a AFC é restrita pela capacidade de seus algoritmos de manipularem bases de dados de larga escala. Porém, tal como exposto em (AUFAURE; GRAND, 2013), apesar dessa limitação, a AFC tem mostrado resultados interessantes na análise de redes sociais, por complementar as métricas tradicionais de centralidade muito comuns no estudo dessas redes.

Assim, este Capítulo apresenta um estudo de caso da rede social *Orkut*. É mostrado que o conjunto de implicações pode ser empregado para caracterização dessas redes. Também é discutido a aplicabilidade da base canônica de implicações na criação de modelos de comunidades.

6.1 Rede Social *Orkut*

O *Orkut* é uma rede social, lançada pelo Google em Janeiro de 2004 com o intuito de permitir que seus usuários pudessem conhecer novas pessoas e manter relacionamentos. No presente trabalho, os dados de acesso dessa rede social, do período de Março de 2009, são utilizados neste estudo de caso. Os dados em questão foram providenciados ao Programa de Pós-Graduação em Informática da PUC Minas, por um provedor de acesso à *Internet*, e devidamente anonimizados. Por questões de privacidade, o nome do provedor de acesso não pode ser revelado.

Uma vez que os dados de acesso fornecidos pelo provedor foram anonimizados, as instâncias de estudo são as sessões de acesso a *Web*. Mais especificamente, somente aquelas relacionadas as páginas do *Orkut*. Assim, uma sessão é definida como o conjunto de transações realizadas em determinado tempo de conexão ao provedor de acesso.

Transações com mesmo identificador de sessão são tratadas como sendo de um mesmo usuário.

Dessa forma, o Contexto Formal, construído a partir dos dados de acesso ao *Orkut*, é definido em termos de sessões. Nesse sentido, os objetos do contexto são definidos como sessões e os atributos correspondem as principais funcionalidades disponíveis nessa rede.

Ao todo, o contexto gerado apresenta 11 atributos, dos quais 1 é multivalorado. Os 10 primeiros atributos desse contexto são referentes as páginas que o usuário, daquela sessão, acessou. Os atributos em questão correspondem as funcionalidades *Home*, *Perfil*, *Álbum*, *Foto*, *Scraps*, *Comunidade*, *FriendAdd*, *Friends*, *Testimonial* e *Logout*.

O último atributo é multivalorado e corresponde ao tempo de início e de fim da transação. Esse atributo se refere ao tempo total da sessão (do usuário), dispondo de 4 opções de marcação. Contudo, como se trata de um atributo multivalorado, apenas 1, das 4 opções disponíveis, é marcada para cada objeto do contexto. Os atributos temporais são definidos nos intervalos *menor do que 30 min*, *entre 30 min e 1 hora*, *entre 1 e 2 horas* e *maior do que 2 horas*.

A Tabela 14 apresenta parte do Contexto Formal construído com os dados de acesso dos usuários do *Orkut*. São mostrados apenas os 5 primeiros objetos desse contexto. À título de simplificação, foram atribuídas letras para o conjunto de atributos desse contexto.

Tabela 14: Parte do Contexto Formal gerado a partir dos dados de acesso dos usários da rede social *Orkut*

	Home (a)	Perfil (b)	Álbum (c)	Foto (d)	Scraps (e)	Comunidade (f)	FriendAdd (h)	Friends (i)	Testimonial (j)	Logout (p)	$<$ 30 min (t)	entre 30 min e 1 hora (u)	entre 1 e 2 horas (v)	$>$ 2 horas (x)
1					×						×			
2					×						×			
3		×									×			
4		×									×			
5				×							×			

Na Tabela 14, 5 dos 64.463 objetos do contexto de entrada representando as sessões dos usuários do *Orkut* são apresentados. Obviamente, como esse contexto apresenta 14 atributos, houve repetição de incidências. Os objetos repetidos foram removidos, de modo que o contexto resultante apresentasse somente ocorrências únicas. Tal pré-processamento resultou na eliminação de 495 objetos. A Figura 10, apresenta o Reticulado de Conceitos

para esse novo contexto.

No Reticulado da Figura 10, é perceptível a divisão dos usuários em 4 (quatro) grupos distintos, agrupados segundo o tempo de duração das sessões. Nessa estrutura, nota-se que os usuários com maior tempo de acesso tendem a fazer uso mais frequente das funcionalidades disponíveis pela plataforma. Isto é corroborado pela presença de nodos (Conceitos) intermediários, comuns aos agrupamentos temporais, e pelo número de arestas ligando os demais nodos aos atributos não-temporais.

Adicionalmente, também foram extraídas o conjunto de implicações do contexto de entrada. Ao todo foram obtidas 242 regras, dentre as quais 157 apresentavam suporte (número de objetos envolvidos) maior do que 0 (zero). Obviamente, como o interesse esta em observar o comportamento dos usuários, regras cujo suporte era nulo foram desconsideradas neste estudo de caso. É importante notar que regras com suporte nulo fazem parte da base canônica de implicações, a qual é completa, não-redundante e mínima. Assim, a eliminação desse tipo de regra é uma mera questão de aplicação.

Através do conjunto de regras extraído foi possível identificar tendências no comportamento dos usuários. Por exemplo, foi identificado que usuários com tempo de acesso superior à 2 horas, permanecem mais tempo na página principal (*Home*) e em perfis de amigos (*Friends*). Nota-se também que esses usuários gastam parte desse tempo em seus próprios perfis e em comunidades. Esses usuários também fazem uso frenquente da funcionalidade *Fotos*.

Outra característica interessante é aquela verificada para os usuários com tempo de acesso inferior a 30 minutos. Nesses casos, nota-se que os usuários acessam a rede social, primordialmente, para postagem de *Scraps*, adicionar um novo amigo ou para acessar os perfis de seus amigos. Essas observações são expressas pelo conjunto de implicações a seguir.

1. {Home, Perfil, Scraps, Friends} $\Rightarrow$ {$>$ 2 horas}
2. {Home, Scraps, Comunidade, Friends} $\Rightarrow$ {$>$ 2 horas}
3. {Foto, Scraps, Comunidade, Friends} $\Rightarrow$ {Home, $>$ 2 horas}
4. {Home, Perfil, Scraps, Comunidade, Friends, $>$ 2 horas} $\Rightarrow$ {Foto}
5. {Scraps, FriendAdd, Friends, $<$ 30 min} $\Rightarrow$ {Perfil}
6. {Perfil, Scraps, FriendAdd, $<$ 30 min} $\Rightarrow$ {Friends}
7. {Perfil, FriendAdd, Friends, $<$ 30 min} $\Rightarrow$ {Scraps}

Num segundo momento, para complementar os resultados observados para essa

rede social, foram criados outros 4 contextos formais, com os mesmos dados de acesso do contexto explicitado anteriormente. Porém, nesse caso, cada um desses 4 contextos representa uma semana do mês de Março. Foram selecionadas apenas as sessões compreendidas no intervalo das 17h às 22h. Esse intervalo corresponde ao horário de pico, ou seja, a faixa de horário em que um maior número de sessões pôde ser observado.

Para cada contexto semanal foram obtidas suas respectivas implicações. A partir daí, foram tomados, dois a dois, as interseções entre os conjuntos gerados. O intuito dessa análise era o de verificar que regras persistiriam, uma vez que isto é um indicativo de que o comportamento dos usários têm se repetido ao longo das semanas, ou seja, um forte indício de formação de hábito. Logo a seguir, é mostrado as implicações obtidas mediante operações de interseção entre as semanas do mês de Março no horário de pico.

- Semana 1 $\bigcap$ Semana 2
  1. {Perfil, Foto, Scraps} $\Rightarrow$ {$>$ 2 horas}
  2. {Testimonial, Logout} $\Rightarrow$ {Home, Perfil, Álbum, Foto, Scraps, Comunidade, FriendAdd, Friends, $>$ 2 horas}
- Semana 1 $\bigcap$ Semana 3
  1. {Perfil, Friends, $>$ 2 horas} $\Rightarrow$ {Home}
  2. {Foto, Scraps, Friends} $\Rightarrow$ {Home, $>$ 2 horas}
  3. {Home, Comunidade, Logout} $\Rightarrow$ {Perfil, Álbum, Foto, Scraps, FriendAdd, Friends, Testimonial, $>$ 2 horas}
  4. {FriendAdd, Logout} $\Rightarrow$ {Home, Perfil, Álbum, Foto, Scraps, Comunidade, Friends, Testimonial, $>$ 2 horas}
  5. {Testimonial, Logout} $\Rightarrow$ {Home, Perfil, Álbum, Foto, Scraps, Comunidade, FriendAdd, Friends, $>$ 2 horas}
- Semana 1 $\bigcap$ Semana 4
  1. {Perfil, Friends, $>$ 2 horas} $\Rightarrow$ {Home}
  2. {Home, Scraps, Friends} $\Rightarrow$ {$>$ 2 horas}
- Semana 2 $\bigcap$ Semana 3
  1. {FriendAdd, Testimonial} $\Rightarrow$ {Home, Perfil, Álbum, Foto, Scraps, Comunidade, Friends, Logout, $>$ 2 horas}

2. {Testimonial, Logout} ⇒ {Home, Perfil, Álbum, Foto, Scraps, Comunidade, FriendAdd, Friends, > 2 horas}

- Semana 2 ⋂ Semana 4

1. {∅}

- Semana 3 ⋂ Semana 4

1. {Álbum, Foto, > 2 horas} ⇒ {Home}
2. {Perfil, Friends, > 2 horas} ⇒ {Home}
3. {Home, Álbum, Foto} ⇒ {> 2 horas}
4. {Home, Perfil, Friends} ⇒ {> 2 horas}

A partir das interseções entre as semanas, percebe-se que apenas as regras correspondentes às sessões com mais de 2 horas de duração são mostradas. Isto faz sentido, já que sessões com esse tempo de duração ocorreram em maior número na base de dados analisada. Logo, é mais provável que existam regras comuns a tais sessões. Essas interseções revelam ainda, características interessantes sobre os usuários. Regras em que todos os atributos são listados, revelam que usuários com tempo de acesso superior à 2 horas, tendem a fazer uso de todas as funcionalidades disponibilizadas pela plataforma. Essas regras são óbvias e pouco informativas.

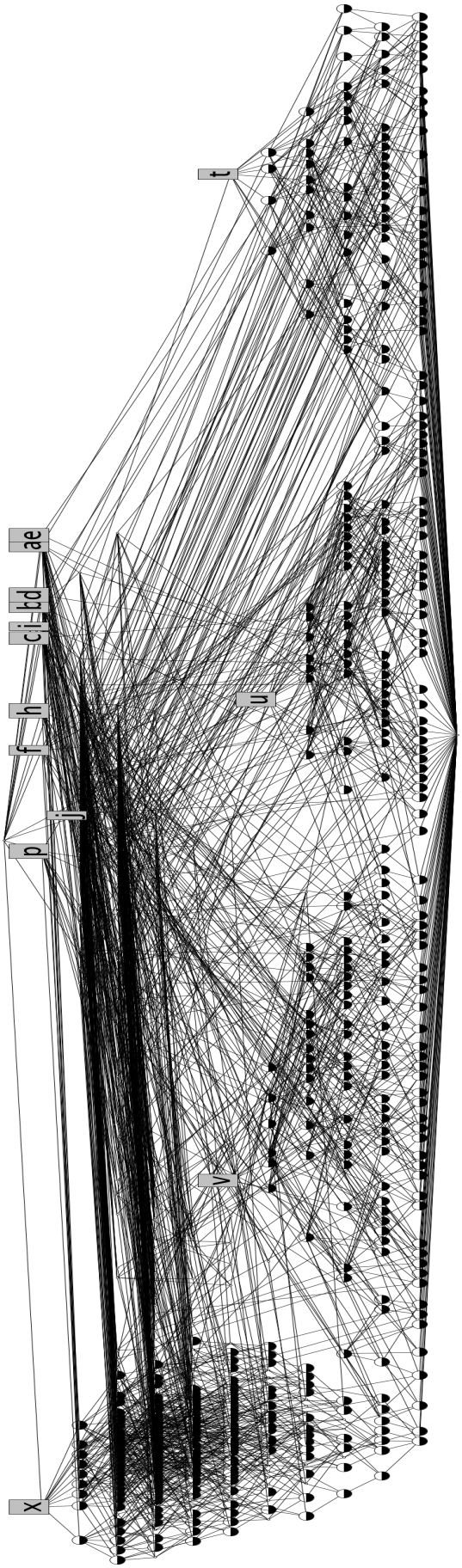
Por outro lado, as demais regras em que não ocorrem todos os atributos são bastante informativas, e dão uma ideia de quais as principais funcionalidades utilizadas pelos usuários. Por razões óbvias, tal como pode ser observado, a principal funcionalidade acessada é a página *Home*, já que é a partir dessa página que os demais recursos são acessados. As demais regras indicam um frequente uso das funcionalidades *Perfil* e *Friends* em conjunto. Também se nota que o recurso *Álbum* só ocorre quando acessada a funcionalidade *Foto*.

De certo, a importância das implicações é evidenciada por sua gama de aplicações. Tais estruturas podem ser empregadas na elaboração de filtros colaborativos, em clusterização ou mesmo na identificação de cópias (COHEN et al., 2001). Ademais, a importância dessas estruturas também é corroborada pelo fato de que a partir do conjunto de implicações é possível reconstruir o Reticulado de Conceitos. Alternativamente, as implicações válidas do contexto de entrada também podem ser obtidas a partir do Reticulado de Conceitos (GANTER; WILLE, 1997). No entanto, o conjunto completo de implicações

válidas de um contexto tende a ser muito grande e a apresentar várias regras triviais. É nesse sentido que a base canônica de implicações se torna de especial interesse, pois esta cobertura é garantidamente a de menor tamanho, não-redundante e completa.

Sobretudo, no tocante a aplicabilidade dessas estruturas, as implicações também podem ser empregadas para a construção de modelos de comunidades. Nesse caso, a construção de modelos de comunidades pode viabilizar a criação de ambientes sintéticos, propícios a realização de simulações. Esses modelos poderiam ser criados de acordo com algum critério estabelecido pelo usuário.

Figura 10: Reticulado Conceitual para o contexto representando as sessões dos usuários ao *Orkut*



7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi discutido o emprego de arquiteturas paralelas como forma de reduzir o alto tempo de execução observado em situações onde o Contexto Formal apresenta elevado número de marcações e alta dimensionalidade. Foi mostrado, por meio dos experimentos, que a paralelização da base canônica de implicações proporciona significativa redução no tempo de execução quando o contexto de entrada apresenta densidade esparsa e elevado número de regras. Entretanto, mesmo com essa redução, os tempos observados ainda são proibitivos quando empregadas densidades superiores a 30% em contextos de larga escala, tal como mostrado na Seção 5.2.

Os experimentos revelam ainda, que o número de *threads* de execução tem significativa influência sobre o desempenho da abordagem paralela. Particularmente, nas simulações em que foram utilizadas 16 *threads* de execução, houve declíneo no *speedup* obtido. Esse comportamento pôde ser observado para todos os contextos considerados nos experimentos, e é justificado, já que essa situação proporciona maior competição entre o time de *threads*, uma vez que o ambiente onde essas simulações foram realizadas dispõe de apenas 8 núcleos de computação.

Em contrapartida, quando empregada uma quantidade igual ou inferior ao número de núcleos disponíveis, houve aumento do *speedup*. Num primeiro momento, tal comportamento pode sugerir que a implementação paralela adotada neste trabalho apresenta melhor desempenho quando o número de *threads* por núcleo é igual a 1. Porém, nas simulações com 16 *threads*, pode-se notar que o aumento do número de atributos no contexto de entrada proporciona melhora no ganho obtido. Ainda que tal desempenho seja inferior em comparação com as simulações com um número inferior de *threads*. Isto mostra que o comportamento observado para 16 *threads* de execução esta relacionado ao tamanho dos grãos.

Também foi mostrada a aplicabilidade da AFC em redes sociais. Em especial, foi realizado um estudo de caso com os dados de acesso dos usuários a rede social *Orkut*, onde foi mostrado que a base canônica de implicações pode ser empregada para caracterizar o comportamento dos usuários dessa rede. Foi também discutido a utilização da base mínima de implicações no modelamento de comunidades, o que pode ser muito útil no

desenvolvimento de ambientes para simulações.

Todavia, neste trabalho somente foram avaliados contextos formais em que o número de objetos é superior ao número de atributos. Não foi avaliado o comportamento do algoritmo para obtenção da base canônica de implicações em situações onde a entrada apresente elevado número de atributos em comparação com o número de objetos. Tampouco foram avaliados contextos onde a dimensionalidade de objetos e de atributos é a mesma.

Em trabalhos futuros, pretende-se avaliar o comportamento de ambas abordagens, sequencial e paralela, nas situações em que o Contexto Formal apresente número de atributos superior ao de objetos e quando a cardinalidade de objetos e de atributos for igual. Também é de interesse avaliar o desempenho da abordagem paralela com as melhorias no algoritmo *NextClosure*, sugeridas por Bazhanov e Obiedkov em (BAZHANOV; OBIEDKOV, 2011), em contextos de larga escala.

REFERÊNCIAS

AUFAURE, M.-A.; GRAND, B. L. **Advances in FCA-based Applications for Social Networks Analysis**. In: *International Journal of Conceptual Structures and Smart Applications (IJCSSA)*. [S.l.: s.n.], 2013. p. 73–89.

BAZHANOV, K.; OBIEDKOV, S. A. Comparing performance of algorithms for generating the duquenne-guigues basis. In: NAPOLI, A.; VYCHODIL, V. (Ed.). *CLA*. [S.l.]: CEUR-WS.org, 2011. (CEUR Workshop Proceedings, v. 959), p. 43–57.

BERRY, A.; BORDAT, J. P.; SIGAYRET, A. **A local approach to concept generation**. *Ann. Math. Artif. Intell.*, v. 49, n. 1-4, p. 117–136, 2007.

BRIN, S.; PAGE, L. **The Anatomy of a Large-Scale Hypertextual Web Search Engine**. *Computer Networks*, v. 30, n. 1-7, p. 107–117, 1998.

CARPINETO, C.; ROMANO, G. ***Concept Data Analysis: Theory and Applications***. [S.l.]: John Wiley & Sons, 2004. ISBN 0470850558.

COHEN, E. et al. **Finding Interesting Associations without Support Pruning**. *IEEE Trans. Knowl. Data Eng.*, v. 13, n. 1, p. 64–78, 2001.

CUVELIER, E.; AUFAURE, M.-A. **A Buzz and E-Reputation Monitoring Tool for Twitter Based on Galois Lattices**. In: ANDREWS, S. et al. (Ed.). *ICCS*. [S.l.]: Springer, 2011. (Lecture Notes in Computer Science, v. 6828), p. 91–103. ISBN 978-3-642-22687-8.

DIAS, S. M. ***Algoritmos para Geração de Reticulados Conceituais***. Dissertação (Mestrado) — Universidade Federal de Minas Gerais (UFMG), Instituto de Ciências Extatas, Departamento de Ciência da Computação, Belo Horizonte, Minas Gerais, Brasil, 2010.

DIAS, S. M.; VIEIRA, N. **Reducing the Size of Concept Lattices: The JBOS Approach**. In: KRYSZKIEWICZ, M.; OBIEDKOV, S. A. (Ed.). *CLA*. [S.l.]: CEUR-WS.org, 2010. (CEUR Workshop Proceedings, v. 672), p. 80–91.

DOERFEL, S.; JÄSCHKE, R.; STUMME, G. **Publication Analysis of the Formal Concept Analysis Community**. In: DOMENACH, F.; IGNATOV, D. I.; POELMANS, J. (Ed.). *ICFCA*. [S.l.]: Springer, 2012. (Lecture Notes in Computer Science, v. 7278), p. 77–95. ISBN 978-3-642-29891-2.

FU, H.; NGUIFO, E. **Partitioning large data to scale up lattice-based algorithm**. *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, IEEE, p. 537–541, 2003.

FU, H.; NGUIFO, E. M. **A Parallel Algorithm to Generate Formal Concepts for Large Data**. In: EKLUND, P. W. (Ed.). *ICFCA*. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 2961), p. 394–401. ISBN 3-540-21043-1.

GANTER, B. **Formal Concepts Analisys: algorithmic aspects**. *TU Dresden, Germany, Tech. Report*, IEEE, 2002.

GANTER, B. **Two Basic Algorithms in Concept Analysis**. In: KWUIDA, L.; SERTKAYA, B. (Ed.). *ICFCA*. [S.l.]: Springer, 2010. (Lecture Notes in Computer Science, v. 5986), p. 312–340. ISBN 978-3-642-11927-9.

GANTER, B.; WILLE, R. ***Formal Concept Analysis: Mathematical Foundations***. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997. Translator-Franzke, C. ISBN 3540627715.

GÉLY, A.; MEDINA, R.; NOURINE, L. **About the Enumeration Algorithms of Closed Sets**. In: KWUIDA, L.; SERTKAYA, B. (Ed.). *ICFCA*. [S.l.]: Springer, 2010. (Lecture Notes in Computer Science, v. 5986), p. 1–16. ISBN 978-3-642-11927-9.

GÉLY, A. et al. **Uncovering and Reducing Hidden Combinatorics in Guigues-Duquenne Bases**. In: GANTER, B.; GODIN, R. (Ed.). *ICFCA*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3403), p. 235–248. ISBN 3-540-24525-1.

GOMIDE, J. S. ***Mineração de Redes Sociais para Detecção e Previsão de Eventos Reais***. Dissertação (Mestrado) — Universidade Federal de Minas Gerais (UFMG), Instituto de Ciências Extatas, Departamento de Ciência da Computação, Belo Horizonte, Minas Gerais, Brasil, 2012.

GUIGUES, J.; DUQUENNE, V. **Familles minimales d'implications informatives résultant, d'un tableau de données binaires**. *Mathématiques et Sciences Humaines*, v. 95, p. 5–18, 1986.

HU, X. et al. **A Parallel Algorithm to Construct Concept Lattice**. *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, IEEE, v. 2, p. 119–123, 2007.

JAY, N.; KOHLER, F.; NAPOLI, A. **Analysis of Social Communities with Iceberg and Stability-Based Concept Lattices**. In: MEDINA, R.; OBIEDKOV, S. A. (Ed.). *ICFCA*. [S.l.]: Springer, 2008. (Lecture Notes in Computer Science, v. 4933), p. 258–272. ISBN 978-3-540-78136-3.

KAYTOUE, M. et al. **Mining Gene Expression Data with Pattern Structures in Formal Concept Analysis**. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 181, n. 10, p. 1989–2001, maio 2011. ISSN 0020-0255. Disponível em: <http://dx.doi.org/10.1016/j.ins.2010.07.007>.

KENGUE, J. F. D.; VALTCHEV, P.; DJAMÉGNI, C. T. **A Parallel Algorithm for Lattice Construction**. In: GANTER, B.; GODIN, R. (Ed.). *ICFCA*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3403), p. 249–264. ISBN 3-540-24525-1.

KENGUE, J. F. D.; VALTCHEV, P.; DJAMÉGNI, C. T. **Parallel Computation of Closed Itemsets and Implication Rule Bases**. In: STOJMENOVIC, I. et al. (Ed.). *ISPA*. [S.l.]: Springer, 2007. (Lecture Notes in Computer Science, v. 4742), p. 359–370. ISBN 978-3-540-74741-3.

KLEINBERG, J. M. **Authoritative Sources in a Hyperlinked Environment**. *J. ACM*, v. 46, n. 5, p. 604–632, 1999.

KRAJCA, P.; OUTRATA, J.; VYCHODIL, V. **Parallel Recursive Algorithm for FCA**. In: *6th Internation Conference on Concept Lattices and Their Applications (CLA 2008)*. Oulomouc, Czech Republic: [s.n.], 2008. v. 433, p. 71–82. ISBN 978-80-244-2111-7.

KRAJCA, P.; OUTRATA, J.; VYCHODIL, V. **Parallel algorithm for computing fixpoints of Galois connections**. *Ann. Math. Artif. Intell.*, v. 59, n. 2, p. 257–272, 2010.

KRAJCA, P.; VYCHODIL, V. **Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework**. In: ADAMS, N. M. et al. (Ed.). *IDA*. [S.l.]: Springer, 2009. (Lecture Notes in Computer Science, v. 5772), p. 333–344. ISBN 978-3-642-03914-0.

KUZNETSOV, S. O. **Learning of Simple Conceptual Graphs from Positive and Negative Examples**. In: ZYTKOW, J. M.; RAUCH, J. (Ed.). *PKDD*. [S.l.]: Springer, 1999. (Lecture Notes in Computer Science, v. 1704), p. 384–391. ISBN 3-540-66490-4.

KUZNETSOV, S. O.; OBIEDKOV, S. A. **Some decision and counting problems of the Duquenne-Guigues basis of implications**. *Discrete Applied Mathematics*, v. 156, n. 11, p. 1994–2003, 2008.

KUZNETSOV, S. O.; OBIEDKOV, S. A.; ROTH, C. **Reducing the Representation Complexity of Lattice-Based Taxonomies**. In: PRISS, U.; POLOVINA, S.; HILL, R. (Ed.). *ICCS*. [S.l.]: Springer, 2007. (Lecture Notes in Computer Science, v. 4604), p. 241–254. ISBN 978-3-540-73680-6.

LI, Y. et al. **Theoretical research on the distributed construction of concept lattices**. In: *Machine Learning and Cybernetics, 2003 International Conference on*. [S.l.: s.n.], 2003. v. 1, p. 474–479.

LINDIG, C.; GBR, G. D. **Fast Concept Analysis**. In: *Working with Conceptual Structures – Contributions to ICCS 2000*. [S.l.]: Shaker Verlag, 2000. p. 152–161.

MORAES, N. R. M. de; ZÁRATE, L. E.; FREITAS, H. C. **A Distributed Algorithm for Formal Concepts Processing based on Search Subspaces**. In: FILIPE, J.; CORDEIRO, J. (Ed.). *ICEIS (1)*. [S.l.]: SciTePress, 2010. p. 105–111. ISBN 978-989-8425-04-1.

NEUHAUS, S.; ZIMMERMANN, T. **The Beauty and the Beast: Vulnerabilities in Red Hat's Packages**. In: *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2009. (USENIX'09), p. 30–30. Disponível em: <http://dl.acm.org/citation.cfm?id=1855807.1855837>.

POELMANS, J. et al. **Formal Concept Analysis in Knowledge Discovery: A Survey**. In: CROITORU, M.; FERRÉ, S.; LUKOSE, D. (Ed.). *ICCS*. [S.l.]: Springer, 2010. (Lecture Notes in Computer Science, v. 6208), p. 139–153. ISBN 978-3-642-14196-6.

PRISS, U. **Formal Concept Analysis in Information Science**. *ARIST*, v. 40, n. 1, p. 521–543, 2006.

PRISS, U. **Unix Systems Monitoring with FCA**. In: ANDREWS, S. et al. (Ed.). *ICCS*. [S.l.]: Springer, 2011. (Lecture Notes in Computer Science, v. 6828), p. 243–256. ISBN 978-3-642-22687-8.

PRISS, U. **Using FCA to Analyse How Students Learn to Program**. In: CELLIER, P.; DISTEL, F.; GANTER, B. (Ed.). *ICFCA*. [S.l.]: Springer, 2013. (Lecture Notes in Computer Science, v. 7880), p. 216–227. ISBN 978-3-642-38316-8, 978-3-642-38317-5.

PRISS, U.; RIEGLER, P.; JENSEN, N. **Using FCA for Modelling Conceptual Difficulties in Learning Processes**. In: DOMENACH, F.; IGNATOV, D.; POELMANS, J. (Ed.). *ICFCA*. [S.l.: s.n.], 2012. p. 161–173.

QI, H. et al. **A parallel algorithm based on search space partition for generating concepts**. *IEEE*, Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on, n. 1, p. 241–248, 2004. ISSN 1521-9097.

RIMSA, A.; ZáRATE, L. E.; SONG, M. A. J. **Handling Large Formal Context Using BDD – Perspectives and Limitations**. In: *Proceedings of the 7th International Conference on Formal Concept Analysis (ICFCA 2009)*. Darmstadt, Germany: Springer-Verlag, 2009. (LNCS/LNAI, v. 5548), p. 194–206. ISBN 978-3-642-01814-5. Disponível em: <http://www.dcc.ufmg.br/∼rimsa/papers/rimsa-icfca.pdf>.

RIMSA, A.; ZáRATE, L. E.; SONG, M. A. J. **SCGaz – A Synthetic Formal Context Generator with Density Control for Test and Evaluation of FCA Algorithms**. 2012.

ROME, J. E.; HARALICK, R. M. **Towards a Formal Concept Analysis Approach to Exploring Communities on the World Wide Web**. In: GANTER, B.; GODIN, R. (Ed.). *ICFCA*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3403), p. 33–48. ISBN 3-540-24525-1.

SILVA, J. P. D. ***Algoritmos de Classificação Baseados em Análise Formal de Conceitos***. Dissertação (Mestrado) — Universidade Federal de Minas Gerais (UFMG), Instituto de Ciências Extatas, Departamento de Ciência da Computação, Belo Horizonte, Minas Gerais, Brasil, 2007.

VALTCHEV, P.; DUQUENNE, V. **Towards Divide-and-Conquer Methods for Computing Concepts and Implications**. In: *Fourth International Conference on Knowledge Discovery and Discrete Mathematics – Journées de l'informatique Messine – JIM'03*. Metz, France: INRIA, 2003. p. 3–15.

VIMIEIRO, R. ***Um Estudo de Algoritmos para a Extração de Regras Baseados em Análise Formal de Conceitos***. Dissertação (Mestrado) — Universidade Federal de Minas Gerais (UFMG), Instituto de Ciências Extatas, Departamento de Ciência da Computação, Belo Horizonte, Minas Gerais, Brasil, 2007.

VYCHODIL, V. **A new algorithm for computing formal concepts**. In: *Proceedings of the 19th European Meeting on Cybernetics and Systems Research (EMCSR 2008)*. [S.l.: s.n.], 2008. p. 15–21. ISBN 978-3-85206-175-7.

WILLE, R. **Restructuring lattice theory: an approach based on hierarchies of concepts.** . In: RIVAL, I. (Ed.). *Ordered sets*. Dordrecht–Boston: Reidel, 1982. p. 445–470.