ALGORITMOS PARA GERAÇÃO DE RETICULADOS CONCEITUAIS.

SÉRGIO MARIANO DIAS
ORIENTADOR: NEWTON JOSÉ VIEIRA

ALGORITMOS PARA GERAÇÃO DE RETICULADOS CONCEITUAIS.

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

Belo Horizonte
Março de 2010

© 2010, Sérgio Mariano Dias.
Todos os direitos reservados.

Dias, Sérgio Mariano
D541a Algoritmos para Geração de Reticulados Conceituais. / Sérgio Mariano Dias. — Belo Horizonte, 2010
xxv, 132 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Newton José Vieira

1. Teoria dos reticulados - Teses. 2. Análise formal de conceitos - Teses. 3. Algoritmos - Teses. I. Orientador. II. Título.

CDU 519.6*22(043)

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

FOLHA DE APROVAÇÃO

Algoritmos para geração de reticulados conceituais

SÉRGIO MARIANO DIAS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. NEWTON JOSÉ VIEIRA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. LUIS ENRIQUE ZÁRATE
Departamento de Ciência da Computação - PUC-MG

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

Belo Horizonte, 26 de março de 2010.

Aos meus pais Dionisio Mariano Dias e Delfina Nunes Neta Dias.

Agradecimentos

Primeiramente gostaria de agradecer o professor Newton José Vieira pela orientação e apoio durante toda elaboração desse trabalho. Agradeço também o professor Luiz Enrique Zárate pelas oportunidades de pesquisa ainda durante minha graduação, período no qual iniciei meus estudos sobre a análise formal de conceitos. E agradeço os colegas de mestrado, em especial: Elton Cardoso e Erica Oliveira.

". . . certain scientific questions can never be answered because the necessary computing resources do not exist in the universe."
(Traub e Wozniakowski [1994])

Resumo

Reticulados conceituais são utilizados na análise formal de conceitos para representação de conhecimento. A análise formal de conceitos teve sua formalização em 1982 com o trabalho apresentado por Wille [1982], que propôs considerar cada elemento de um reticulado como um conceito formal e o reticulado como uma hierarquia das relações entre os conceitos.

Atualmente existe um crescente interesse em aplicações da análise formal de conceitos em diversas áreas da ciência. Entretanto, o reticulado conceitual pode apresentar um comportamento (número de conceitos formais) exponencial. Apesar desse comportamento ser raramente encontrado em casos práticos, ainda assim o custo computacional pode ser proibitivo para muitas aplicações, criando assim a necessidade de novos algoritmos para geração dos reticulados conceituais e abordagens para redução de sua complexidade.

Esse trabalho apresenta duas propostas para redução da complexidade dos reticulados conceituais e discute uma metodologia para avaliação dessas propostas. Propõe um arcabouço didático para o desenvolvimento de algoritmos da análise formal de conceitos chamado *EF-Concept Analysis* (*Educational Framework for Concept Analysis*) e apresenta um *benchmark* para avaliação dos paradigmas de construção dos reticulados conceituais, o qual pode ser expandido e utilizado pela comunidade científica.

Abstract

Concept lattices are used in formal concept analysis for knowledge representation. Formal concept analysis was born in 1982 with the work presented by Wille [1982], who proposed to consider each lattice element as a formal concept and the lattice as a hierarchy of relationships between concepts.

Currently there is a growing interest in formal concept analysis applications in many science areas. However, the concept lattice can behave (number of formal concepts) exponentially. Although this behavior is rarely found in practical cases, yet the computational cost can be prohibitive for many applications, thus creating the need for new algorithms to build concept lattices and approaches to reduce this complexity.

This paper presents two proposals for reducing the concept lattices complexity and discusses a methodology for evaluating such proposals. It proposes an educational framework to develop algorithms for formal concepts analysis called *EF-Concept Analysis* (*Educational Framework for Concept Analysis*) and present a *benchmark* for evaluation of constructing concept lattices paradigms, which can be expanded and used by the scientific community.

Sumário

1 Introdução **1**

1.1 Problema 3

1.2 Objetivos 4

1.3 Organização da Dissertação 5

2 Análise Formal de Conceitos: revisão de conceitos **7**

2.1 Ordens Parciais 7

2.1.1 Conjuntos Ordenados 7

2.1.2 Diagrama de Linhas 8

2.2 Sistemas de Fecho 10

2.3 Análise Formal de Conceitos 12

2.3.1 Contextos Formais 12

2.3.1.1 Contextos Formais Multivalorados 14

2.3.2 Conceitos Formais 14

2.3.3 Reticulados Conceituais 15

3 Algoritmos **17**

3.1 Complexidade do Reticulado Conceitual 17

3.2 O Processo de Construção do Reticulado Conceitual 19

3.3 Gerando Conceitos Formais 20

3.3.1 Algoritmo Next Closure 23

3.4 Obtendo o Reticulado Conceitual a partir dos Conceitos Formais 25

3.4.1 Algoritmo Concepts Cover 25

3.5 Obtendo o Reticulado Conceitual e os Conceitos Formais 27

3.5.1 Algoritmo Next Neighbours 27

3.5.2 Algoritmos Incrementais 29
3.5.2.1 Algoritmo Godin 29
3.6 Abordagens Alternativas para Geração dos Conceitos Formais 30
3.7 Comparação entre Algoritmos 33
3.8 Alternativas para Redução da Complexidade 35
3.8.1 Redução Baseada no Contexto Formal. 36
3.8.2 Redução Baseada no Reticulado Conceitual. 38

4 Um Arcabouço para Desenvolvimento e Avaliação de Algoritmos da AFC 43
4.1 O Arcabouço EF-Concept Analysis 43
4.2 Benchmark para Avaliação dos Algoritmos e Paradigmas 46
4.2.1 Benchmark Sintético 48
4.2.2 Benchmark Real 51
4.2.2.1 Base de Dados Íris 52
4.2.2.2 Base de Dados Wine 54
4.2.2.3 Base de Dados Yeast 55
4.2.2.4 Base de Dados Water 56
4.2.2.5 Resumo das Características das Bases de Dados Reais 56

5 Redução da Complexidade do Reticulado Conceitual 59
5.1 Avaliação das Técnicas de Redução da Complexidade 59
5.1.1 Um Algoritmo para Extração de Regras de Implicação. 64
5.2 Propostas de Redução do Reticulado Conceitual 67
5.2.1 Junção Baseada em Similaridade - JBS 67
5.2.2 Representação Baseada em Agrupamentos - RBA 73

6 Análise dos Resultados 79
6.1 Avaliações do Benchmark Proposto 80
6.1.1 Avaliações do Benchmark Sintético 80
6.1.2 Avaliações do Benchmark Real 88
6.2 Avaliação dos Métodos de Redução da Complexidade 90
6.2.1 Avaliação do Método de Junção Baseada em Similaridade . . . 90
6.2.2 Avaliação do Método de Representação Baseada em Agrupamentos 98

7 Conclusões **103**

A Informações Complementares das Bases de Dados Reais **109**

B Informações Complementares da Avaliação do Método de Junção Baseada em Similaridade **115**

Referências Bibliográficas **125**

Lista de Figuras

1.1 Exemplo de diagrama de linhas para taxonomia de animais. 3

2.1 Diagrama de linhas do conjunto potência de {a,b,c}. 9

3.1 Reticulado conceitual frequente para um suporte mínimo de 33%. 40

4.1 Diagrama de pacotes do arcabouço *EF-Concept Analysis*. 45
4.2 Diagrama de classe com as principais estruturas e algoritmos. 46
4.3 Diagrama de linhas para base de dados Íris. 54

5.1 Grau de equivalência entre o contexto formal original e um conjunto de regras de implicação. 62
5.2 Diagrama de linhas após aplicação do método JBS. 71
5.3 Reticulado conceitual criado pelo método RBA para r=1. 75
5.4 Reticulado conceitual criado pelo método RBA para r=2. 75
5.5 Reticulado conceitual criado pelo método RBA para r=3. 75

6.1 Número de conceitos para $|G| = 20$, $|M| = 50$ e uma distribuição aleatória. . 82
6.2 Tempo em milissegundos para $|G| = 20$, $|M| = 50$ e uma distribuição aleatória. 82
6.3 Número de conceitos para $|G| = 50$, $|M| = 20$ e uma distribuição aleatória. . 83
6.4 Tempo em milissegundos dos paradigmas para $|G| = 50$, $|M| = 20$ e uma distribuição aleatória. 83
6.5 Número de conceitos para $|G| = 20$, $|M| = 50$ e uma densidade média. . . . 85
6.6 Tempo em milissegundos dos paradigmas para $|G| = 20$, $|M| = 50$ e uma densidade média. 85
6.7 Número de conceitos para $|G| = 50$, $|M| = 20$ e uma densidade média. . . . 86

6.8 Tempo em milissegundos dos paradigmas para $|G| = 50$, $|M| = 20$ e uma densidade média. 86
6.9 Número de conceitos para $|M| = 20$, $|I| = 50\%$ aleatória e uma variação do número de objetos. 87
6.10 Tempo em milissegundos dos paradigmas para $|M| = 20$, $|I| = 50\%$ aleatória e uma variação do número de objetos. 87
6.11 Número de conceitos para $|G| = 20$, $|I| = 50\%$ aleatória e uma variação do número de atributos. 88
6.12 Tempo em milissegundos dos paradigmas para $|G| = 20$, $|I| = 50\%$ aleatória e uma variação do número de atributos. 88
6.13 Número de objetos, densidade, número de conceitos formais e número de regras obtidos para a base Íris aplicada no método JBS. 92
6.14 Fidelidade média e perda descritiva para a base de dados Íris aplicada no método JBS. 93
6.15 Número de objetos, densidade, número de conceitos formais e número de regras obtidos para a base Wine aplicada no método JBS. 94
6.16 Fidelidade média e perda descritiva para a base de dados Wine aplicada no método JBS. 94
6.17 Número de objetos, densidade, número de conceitos formais e número de regras obtidos para a base Yeast aplicada no método JBS. 95
6.18 Fidelidade média e perda descritiva para a base de dados Yeast aplicada no método JBS. 95
6.19 Número de objetos, densidade e número de conceitos formais para a base Water aplicada no método JBS. 97
6.20 Fidelidade média e perda descritiva para a base de dados Water aplicada no método JBS. 97

A.1 Distribuição do atributo “sepal length (cm)” para a base de dados Iris. . . . 109
A.2 Distribuição do atributo “sepal width (cm)” para a base de dados Iris. 109
A.3 Distribuição do atributo “petal length (cm)” para a base de dados Iris. . . . 109
A.4 Distribuição do atributo “petal width (cm)” para a base de dados Iris. 109
A.5 Distribuição das classes para a base de dados Iris. 110
A.6 Distribuição do atributo “Alcohol” para a base de dados Wine. 110

A.7 Distribuição do atributo “Malic acid” para a base de dados Wine. 110
A.8 Distribuição do atributo “Ash” para a base de dados Wine. 110
A.9 Distribuição do atributo “Alcalinity of ash” para a base de dados Wine. . . . 110
A.10 Distribuição do atributo “Magnesium” para a base de dados Wine. 111
A.11 Distribuição do atributo “Total phenols” para a base de dados Wine. 111
A.12 Distribuição do atributo “Flavanoids” para a base de dados Wine. 111
A.13 Distribuição do atributo “Nonflavanoid phenols” para a base de dados Wine. 111
A.14 Distribuição do atributo “Proanthocyanins” para a base de dados Wine. . . . 111
A.15 Distribuição do atributo “Color intensity” para a base de dados Wine. 111
A.16 Distribuição do atributo “Hue” para a base de dados Wine. 112
A.17 Distribuição do atributo “OD280/OD315 of diluted wines” para a base de dados Wine. 112
A.18 Distribuição do atributo “Proline” para a base de dados Wine. 112
A.19 Distribuição das classes para a base de dados Wine. 112
A.20 Distribuição do atributo “MCG” para a base de dados Yeast. 112
A.21 Distribuição do atributo “GVH” para a base de dados Yeast. 112
A.22 Distribuição do atributo “ALM” para a base de dados Yeast. 113
A.23 Distribuição do atributo “MIT” para a base de dados Yeast. 113
A.24 Distribuição do atributo “ERL” para a base de dados Yeast. 113
A.25 Distribuição do atributo “POX” para a base de dados Yeast. 113
A.26 Distribuição do atributo “VAC” para a base de dados Yeast. 113
A.27 Distribuição do atributo “NUC” para a base de dados Yeast. 113
A.28 Distribuição das classes para a base de dados Yeast. 114

Lista de Tabelas

1.1 Exemplo de contexto formal para taxonomia de animais. 2

2.1 Obtendo um contexto univalorado a partir de um contexto multivalorado. . . 14

3.1 Exemplo de contexto formal para o pior caso. 18

4.1 Atributo criado para o contexto formal, intervalos e incidência para base de dados Íris. 53
4.2 Atributo criado para o contexto formal, intervalos e incidência para base de dados Wine. 55
4.3 Atributo criado para o contexto formal, intervalos e incidência para base de dados Yeast. 57
4.4 Resumo das características das bases de dados reais que compõem o *benchmark*. 58
4.5 Resumo das características das bases de dados reais que compõem o *benchmark* após redução. 58

5.1 Conjunto de regras de implicação do reticulado conceitual para o exemplo dos animais. 65
5.2 Peso aplicado ao objetos para o exemplo sobre taxonomia dos animais. . . . 70
5.3 Matriz de disparidade obtida para o exemplo sobre taxonomia dos animais. . 70
5.4 Contexto formal para taxonomia de animais após aplicação do método JBS. 70
5.5 Matriz de disparidade obtida para o exemplo sobre taxonomia dos animais utilizando um peso igual a 1 em todos os atributos. 72
5.6 Fidelidade e Perda descritiva para o contexto formal descrevendo a taxonomia dos animais. 76

6.1 Desempenho dos paradigmas de construção do reticulado conceitual para o pior caso. 81
6.2 Aplicações do *benchmark* real nos paradigmas de construção do reticulado conceitual. 89
6.3 Base Íris aplicada no método RBA. 99
6.4 Base Wine aplicada no método RBA. 100
6.5 Base Yeast aplicada no método RBA. 101
6.6 Base Water aplicada no método RBA. 101

7.1 Resumo cronológico dos principais algoritmos. 104

B.1 Base Íris aplicada no método JBS. Parte A 116
B.2 Base Íris aplicada no método JBS. Parte B 117
B.3 Base Wine aplicada no método JBS. Parte A 118
B.4 Base Wine aplicada no método JBS. Parte B 119
B.5 Base Yeast aplicada no método JBS. Parte A 120
B.6 Base Yeast aplicada no método JBS. Parte B 121
B.7 Base Water aplicada no método JBS. Parte A 122
B.8 Base Water aplicada no método JBS. Parte B 123

Lista de Algoritmos

1 Função Next Set 21
2 Algoritmo List of Concepts 21
3 Algoritmo Intersection Set 22
4 Função Next Closure 24
5 Algoritmo All Concepts 24
6 Algoritmo Concepts Cover 26
7 Algoritmo Next Neighbours 28
8 Função Find Lower Neighbours 28
9 Algoritmo Godin 31
10 Algoritmo Find Implications 65
11 Função Find Implications From Concept 66
12 Função Agrupar Objetos 69
13 Algoritmo RBA 74

Capítulo 1

Introdução

A análise formal de conceitos (AFC)[1] é uma técnica da matemática aplicada, baseada na matematização da noção de conceito e na hierarquia conceitual[Ganter e Wille, 1999]. Sua formalização teve início em 1982 com o trabalho apresentado por Wille [1982], que propôs considerar cada elemento de um reticulado como um conceito formal e o reticulado como uma hierarquia das relações entre os conceitos, criando assim, a partir dessa perspectiva, uma hierarquia conceitual. Pode-se dizer que a AFC foi inspirada na necessidade/desejo da formalização da noção filosófica de conceito, que, segundo a filosofia, é um pré-requisito para a formação de opinião e a capacidade de argumentação [Wille, 2005].

Na análise formal relações binárias são representadas através de diagramas, o que permite a organização do conhecimento de uma forma estruturada e organizada hierarquicamente. Atualmente existe um crescente interesse em aplicações da AFC na recuperação de informação [Carpineto e Romano, 2004b,a], mineração de dados [Kovács, 2004], engenharia de software [Buchli, 2003], redes neurais [Zárate et al., 2008; Tsopze et al., 2008, 2007], redes sociais [Jay et al., 2008], ontologias [Stumme, 2002], etc.

Segundo Stumme [2002], nos últimos anos a AFC tem migrado cada vez mais de pesquisas teóricas, apresentadas em congressos matemáticos, para pesquisas práticas apresentadas principalmente em congressos da área de computação. Outro trabalho que vem reforçar essa tendência é o livro [Carpineto e Romano, 2004a], que apresenta a AFC de uma forma prática e voltada para a ciência da computação.

[1] Do inglês, *formal concept analysis*. Será utilizado no texto análise formal de conceitos, análise formal ou simplesmente AFC.

Na AFC o contexto formal é uma definição primordial e pode ser representado por meio de tabelas[2]. Na Tabela 1.1, um exemplo de contexto formal para a taxonomia de animais é apresentado. Essa tabela representa uma estrutura que define objetos (animais), atributos (aspectos dos animais) e seus relacionamentos (que animais tem que aspecto). Cada linha dessa tabela representa um animal: (1) *jacaré*, (2) *sapo*, (3) *humano*, (4) *macaco*, (5) *coruja*, (6) *tubarão*. Cada coluna representa uma característica: (a) *aquático*, (b) *terrestre*, (c) *respiração branquial*, (d) *respiração pulmonar*, (e) *pêlos*, (f) *penas*, (g) *glândulas mamárias*, (h) *raciocínio*. E cada marcação ($\times$) representa o relacionamento entre um objeto e um atributo.

Tabela 1.1: Exemplo de contexto formal para taxonomia de animais.

Objetos	**Atributos**							
	a	b	c	d	e	f	g	h
1		$\times$		$\times$				
2	$\times$	$\times$	$\times$	$\times$				
3		$\times$		$\times$	$\times$		$\times$	$\times$
4		$\times$		$\times$	$\times$		$\times$	
5		$\times$		$\times$		$\times$		
6	$\times$		$\times$					

Um dos aspectos mais poderosos da AFC é sua capacidade gráfica de representar tabelas por meio de diagramas de linhas[3]. Na Figura 1.1, o diagrama de linhas para o exemplo de taxonomia de animais é apresentado. Nele cada objeto é anotado na parte inferior de um nó, e cada atributo na parte superior de um nó. Percorrendo-se o diagrama de baixo para cima a apartir de um nó com certo objeto, obtém-se seus atributos. Simetricamente, percorrendo-se o diagrama de cima para baixo a partir de um nó com certo atributo, obtém- se os objetos que os contêm.

A análise dos diagramas em vez das tabelas é certamente mais intuitiva. Por exemplo, analisando a Figura 1.1 percebe-se facilmente que uma característica que distingue o "*humano*" do "*macaco*" é a capacidade de "*raciocínio*" e que uma característica que distingue o "*sapo*" do "*tubarão*" é o fato do primeiro ser "*terrestre*". Uma análise similar pode ser feita para todos os objetos em estudo.

[2]Em Inglês é utilizado o termo *cross table*.

[3]Um diagrama de linhas é uma forma de representar uma hierarquia conceitual.

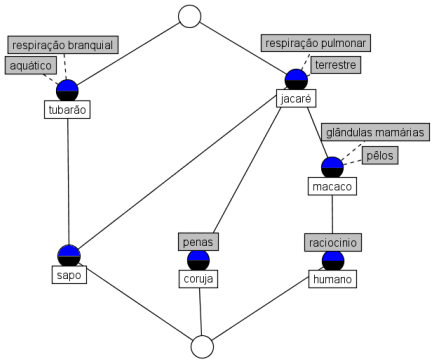

Figura 1.1: Exemplo de diagrama de linhas para taxonomia de animais.

Outra característica possível de se obter analisando a Figura 1.1 é a implicação entre atributos. Pode-se afirmar com 100% de confiança, para os dados em análise, que todo animal com "*pêlos*" é "*terrestre*", ou seja, **Se** { *pêlos* } **Então** { *terrestre* }. Novamente, esse tipo de inferência pode ser aplicado a todos os atributos em estudo.

Cabe ressaltar, que a utilização da análise formal não é restrita unicamente a uma interpretação visual do diagrama de linhas. Em muitas aplicações esse diagrama é a estrutura central da aplicação, fornecendo/mantendo a informação estruturada e organizada.

1.1 Problema

O grande potencial da AFC está na organização do conhecimento em uma hierarquia conceitual, e, de fato, as principais aplicações fazem uso dessa hierarquia representada por diagrama de linhas, diagrama de linhas aninhado, árvores, etc.

Contudo, gerar todos os conceitos formais e classificá-los hierarquicamente no pior caso apresenta um comportamento exponencial [Kuznetsov, 2001]. Apesar desse comportamento ser raramente encontrado em casos práticos [Godin et al., 1986], ainda assim o custo computacional pode ser proibitivo para muitas aplicações [Carpineto e Romano, 2004a]. Em Old e Priss [2006] são apresentados alguns problemas em aberto na análise formal; dentre eles, destaca-se: "como calcular/gerar todos os conceitos formais de um contexto muito grande (120 000 objetos e 70 000 atributos)"?

Desta forma surge a necessidade da proposta de novos algoritmos e abordagens

para geração dos conceitos formais. De fato, são encontrados diversos algoritmos na literatura, como será apresentado no capítulo 3; entretanto, muitos são melhorias dos algoritmos mais conhecidos, criando assim duas outras perguntas importantes: 1) qual algoritmo é mais adequado para gerar os conceitos e o reticulado conceitual, e 2) como comprovar a eficiência de um novo algoritmo/abordagem?

Novamente, alguns trabalhos na literatura tentam responder essas duas últimas perguntas realizando comparações empíricas entre os diversos algoritmos disponíveis (Capítulo 3). Entretanto, vários problemas são detectados nesses trabalhos: a utilização de instâncias de testes inadequadas (pequenas, aleatórias, não representativas), implementações ineficientes de algoritmos, falta de descrição das estruturas de dados utilizadas, interpretações errôneas dos algoritmos previamente apresentados na literatura, dificuldades na identificação do comportamento dos algoritmos, inexistência de uma padronização dos testes, etc.

Portanto, faz-se necessário o estudo dos algoritmos para geração dos conceitos formais e de sua organização hierárquica, a proposta de novos algoritmos, representações para os conceitos formais, seleção de conceitos mais representativos e até mesmo a aplicação de heurísticas para geração parcial do reticulado. Em resumo, é necessária a proposta de novas abordagens capazes de diminuir a complexidade de espaço e tempo desses algoritmos mantendo a qualidade da informação representada.

1.2 Objetivos

Este trabalho possui como objetivos apresentar abordagens para reduzir a complexidade de reticulados conceituais, propor uma metodologia para avaliação dessas abordagens e propor uma padronização para os testes de novos algoritmos para geração de reticulados conceituais. Para alcançar tais objetivos foram estudados os principais algoritmos para geração dos reticulados conceituais e métodos para redução da complexidade dos reticulados conceituais.

Dessa forma, o trabalho pretende apresentar três contribuições principais:

- propor técnicas capazes de reduzir a complexidade do reticulado conceitual;
- propor uma metodologia de avaliação das técnicas de redução da complexidade dos reticulados conceituais;

- propor um *benchmark*, isto é, um conjunto de contextos formais bem definidos que possa ser utilizado pela comunidade para avaliação e comparação de novos algoritmos/abordagens para geração de conceitos formais e sua ordenação no reticulado conceitual.

São objetivos específicos:

- identificar os principais métodos encontrados na literatura para geração de todos os conceitos formais de um contexto formal e sua ordenação em um reticulado conceitual;
- desenvolver um arcabouço didático para a análise formal de conceitos que possibilite a implementação e teste dos algoritmos e metodologias propostas no trabalho;
- implementar os principais algoritmos encontrados na literatura;
- estabelecer um padrão de contextos artificiais e reais que possam ser utilizados para a comparação experimental do desempenho dos algoritmos implementados;
- avaliar os algoritmos implementados utilizando o padrão de testes proposto;
- analisar a qualidade do padrão de testes proposto;
- identificar as principais abordagens para redução da complexidade do reticulado conceitual observadas na literatura;
- propor novas abordagens para redução da complexidade do reticulado;
- propor uma metodologia para avaliação das técnicas de redução da complexidade do reticulado conceitual;
- analisar a metodologia proposta verificando assim a qualidade e viabilidade da mesma.

1.3 Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma:

- o 2º capítulo apresenta uma revisão da análise formal e da teoria necessária para uma melhor compreensão do texto;
- o 3º capítulo apresenta uma revisão bibliográfica dos algoritmos encontrados na literatura, dos trabalhos que comparam esses algoritmos e das abordagens que procuram reduzir a complexidade do processo de gerar todos os conceitos formais e ordená-los hierarquicamente;
- o 4º capítulo discute as propostas deste trabalho para o desenvolvimento dos algoritmos da análise formal e apresenta um *benchmark* para avaliação dos algoritmos;
- o 5º capítulo discute um método para avaliação das técnicas de redução da complexidade do reticulado conceitual e apresenta novas propostas para redução da complexidade do reticulado conceitual;
- o 6º capítulo avalia os paradigmas de construção do reticulado conceitual apresentados utilizando os *benchmark* propostos. Além disso, as técnicas de redução da complexidade do reticulado conceitual propostas são avaliadas utilizando o método também proposto neste trabalho;
- finalmente, o Capítulo 7 apresenta as conclusões do trabalho, assim como as possíveis direções para trabalhos futuros.

Capítulo 2

Análise Formal de Conceitos: revisão de conceitos

Este capítulo apresenta os principais conceitos sobre a análise formal de conceitos, que foi proposta por Wille [1982] como uma técnica da matematização da noção de um conceito. Como a AFC é fundamentada na teoria dos reticulados completos [Davey e Priestley, 1990] os conceitos de ordens e reticulados precisam ser primeiramente apresentados. Em seguida, a definição de sistemas de fecho é feita e, por último, os conceitos da AFC são discutidos.

O leitor interessado no aprofundamento dos conceitos abordados, visto que será apresentada apenas uma rápida revisão de conceitos, pode consultar os livros [Davey e Priestley, 1990; Ganter e Wille, 1999; Axler e Ribet, 2005], além das referências indicadas no texto.

2.1 Ordens Parciais

2.1.1 Conjuntos Ordenados

Primeiro, segundo, terceiro, ..., décimo. Maior, menor, igual. A noção de ordem surge naturalmente quando se deseja comparar objetos de um conjunto.

Em termos matemáticos, uma ordenação, i.e. uma ordem, é uma relação binária aplicada em um conjunto de objetos. Ela não é uma propriedade intrínseca a um único objeto, mas à comparação entre pares de objetos. Segue a definição formal de uma

ordem:

Definição 1. *Seja P um conjunto. Uma ordem (ou ordem parcial) em P é uma relação binária $\leqslant$ em P tal que, para todo x, y, z $\in$ P tem-se:*

- $x \leqslant x$;
- $x \leqslant y$ *e* $y \leqslant x$ *implica* $x = y$;
- $x \leqslant y$ *e* $y \leqslant z$ *implica* $x \leqslant z$.

Essas condições afirmam, respectivamente, que a relação é *reflexiva*, *anti-simétrica* e *transitiva* [Davey e Priestley, 1990; Nation, 200; Harzheim, 2005; Axler e Ribet, 2005]. Um conjunto P com uma ordem $\leqslant$, $(P;\leqslant)$, é chamado conjunto ordenado ou conjunto parcialmente ordenado[1]. Se a relação $\leqslant$ satisfizer uma quarta propriedade chamada *comparabilidade* (para todo $x, y \in P$, $x \leqslant y$ ou $y \leqslant x$), então ela é uma *ordem total*.

Como exemplo de *ordem total*, pode-se citar a relação de ordem "menor ou igual" $\leqslant$, imposta aos números naturais N. Já como exemplo de *ordem parcial* (não necessáriamente total) pode-se citar a relação subconjunto em uma família de conjuntos. Seja X um conjunto e $\wp(X)$ o conjunto potência de X, definido por $\wp(X) = \{Y | Y \subseteq X\}$. Para A e $B \in \wp(X)$ tem-se $A \leqslant B$ se, somente se, $A \subseteq B$, que é chamada de ordem/ordenação pela inclusão. Esse exemplo é utilizado na comparação de conceitos formais como será demonstrado nas próximas subseções.

Dada uma ordem parcial $(P;\leqslant)$, será usada a notação $x < y$ para significar que $x \leqslant y$ e $x \neq y$; e será usada a notação $y \geqslant x$ para significar $x \leqslant y$.

2.1.2 Diagrama de Linhas

Dado um conjunto ordenado finito P, pode-se representá-lo por um desenho, que descreve como os elementos de P estão relacionados, criando assim uma relação de cobertura entre os elementos dessa ordem [Davey e Priestley, 1990].

Definição 2. *Seja uma ordem parcial $(P;\leqslant)$ e x, y $\in$ P. Então x é coberto por y, $x \prec y$, se, somente se, $x < y$ e $x \leqslant z < y$ implica em $z = x$.*

[1] Do Inglês *ordered set* ou *partially ordered set*.

Pode-se entender a definição de cobertura da seguinte forma: $x \prec y$ se não existe um elemento z entre x e y. Nesse caso, x é chamado vizinho inferior e y vizinho superior.

A relação de cobertura dos elementos de um conjunto ordenado P pode ser representada por um grafo, onde cada vértice representa um elemento do conjunto e as arestas relacionamentos entre os elementos. Se ao desenhar o grafo todo relacionamento $x \prec y$ for representado visualmente com o vértice x inferior ao vértice y, então tem-se um diagrama de linhas[2].

Na Figura 2.1 o diagrama do conjunto potência do conjunto $\{a,b,c\}$ é apresentado; observe que para o exemplo tem-se $A \prec B$ se, somente se, $B = A \cup \{e\}$ para $e \in \{a,b,c\} - A$.

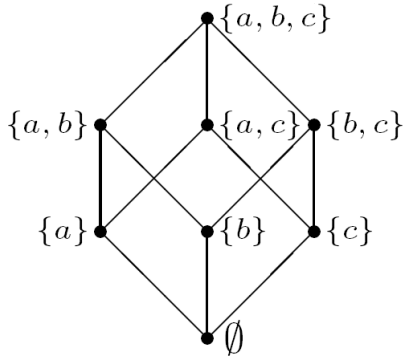

Figura 2.1: Diagrama de linhas do conjunto potência de {a,b,c}.

Associado a um conjunto ordenado P têm-se os conceitos de limite inferior e superior. Primeiramente, para efeitos de definição, considere x como o menor elemento do conjunto ordenado $(P, \leqslant)$ se, somente se, $\forall y \in P \; x \leqslant y$. Dualmente, considere x o maior elemento, se, e somente se, $\forall y \in P \; x \geqslant y$.

Definição 3. *Seja P um conjunto ordenado e $S \subseteq P$. Um elemento $x \in P$ será um limite superior de S se para todo $s \in S$ tem-se $s \leqslant x$. Já para o dual, um elemento $x \in P$ será um limite inferior de S se para todo $s \in S$ tem-se $x \leqslant s$.*

O conjunto de todos os limites superiores de S é denotado por S^u ($S^u = \{x \in P | (\forall s \in S) s \leqslant x\}$). O dual, isto é, o conjunto de todos os limites inferiores será denotado por S^l ($S^l = \{x \in P | (\forall s \in S) x \leqslant s\}$). Se S^u possui um menor elemento, então esse elemento

[2]Na literatura também é utilizado o termo diagrama de Hasse, do inglês *Hasse diagram*.

é chamado menor limite superior[3] de S. Dualmente, se S^l possui um maior elemento, então esse elemento será o maior limite inferior[4] de S. O menor limite superior e o maior limite inferior também são chamados de supremo e ínfimo, respectivamente. A notação $\bigvee S$ para o supremo e $\bigwedge S$ para o ínfimo também são utilizadas; note que nesse caso se $S = \{x, y\}$, então adota-se $x \vee y$ e $x \wedge y$ [Davey e Priestley, 1990].

A partir dos conceitos de supremo e ínfimo, pode-se definir o que é um reticulado completo:

Definição 4. *Seja P um conjunto ordenado não vazio.*

- *Se $x \vee y$ e $x \wedge y$ existem para quaisquer $x, y \in P$, então P será chamado reticulado.*
- *Se $\bigvee S$ e $\bigwedge S$ existem para todo $S \subseteq P$, então P será chamado reticulado completo.*

Note que, o diagrama construído para os elementos do conjunto potência (Figura 2.1) é um reticulado completo, em que as operações para determinação do ínfimo e supremo são realizadas através da união e interseção dos conjuntos. De forma semelhante, a análise formal de conceitos utiliza um reticulado completo, construído sobre conjuntos fechados sob interseções, para representar a noção abstrata de conceitos.

2.2 Sistemas de Fecho

Um segundo grupo de definições importantes são as de: *sistemas de fecho*, *operadores de fecho* e *conexão de Galois*. Esses conceitos são relacionados com os operadores de derivação e com o reticulado conceitual.

Sistemas de fecho[5] são conjuntos de subconjuntos com algumas propriedades. Segue a definição formal:

Definição 5. *Seja $\wp(X)$ o conjunto potência de X. Um sistema de fecho do conjunto X é um subconjunto $C \subseteq \wp(X)$ que satisfaz [Ganter e Stumme, 2003]:*

- *$X \in C$, e*

[3] Do Inglês: *least upper bound*.
[4] Do Inglês: *greatest lower bound*.
[5] Do Inglês: *Closure systems*.

- *Se $D \subseteq C$, então $\bigcap D \in C$.*

O conjunto C é chamado sistema de fecho já que ele é fechado sob interseção, ou seja, a interseção de dois conjuntos provenientes de C resulta em um conjunto de C.

Um sistema de fecho ordenado $(C, \subseteq)$ é um reticulado completo em que o supremo e o ínfimo são obtidos da seguinte forma:

$$\bigvee D = \bigcap \{Y \in C | \bigcup D \subseteq Y\} \tag{2.1}$$

$$\bigwedge D = \bigcap D \tag{2.2}$$

Associado ao conceito de sistemas de fecho existe o conceito de operador de fecho, que é uma função de mapeamento entre dois conjuntos. Um operador de fecho φ é definido da seguinte forma:

Definição 6. *Seja H um conjunto. Uma função φ: $\wp(H) \rightarrow \wp(H)$ é operador de fecho sobre H se para todo $A, B \subseteq H$:*

- $A \subseteq \varphi(A)$;
- *se $A \subseteq B$, então $\varphi(A) \subseteq \varphi(B)$;*
- $\varphi(\varphi(A)) = \varphi(A)$.

Essas propriedades que definem um operador de fecho são chamadas de *monotonicidade*, *extensibilidade* e *idempotência*, respectivamente.

Operadores de fecho e reticulados também estão relacionados. A partir de um operador de fecho pode se obter um reticulado. Seja φ um operador de fecho definido sobre um conjunto X. O conjunto $L = \{C \subseteq X | C = \varphi(C)\}$ ordenado pela inclusão de conjuntos é um reticulado completo em que o ínfimo e o supremo são obtidos por $\bigwedge Y = \bigcap Y$ e $\bigvee Y = \varphi(\bigcup Y)$, respectivamente [Ganter e Wille, 1998; Davey e Priestley, 1990].

Um último conceito importante é o de conexão de Galois; segue sua definição.

Definição 7. *Sejam $(P, \leqslant)$ e $(Q, \leqslant)$ dois conjuntos ordenados, $p, p1, p2 \in P$, $q, q1, q2 \in Q$ e φ, ψ dois operadores de fecho, em que φ: $P \rightarrow Q$ e ψ: $Q \rightarrow P$. Então, o sistema criado entre P e Q será um conexão de Galois se:*

- $p1 \leqslant p2 \Rightarrow \varphi p1 \geqslant \varphi p2$
- $q1 \leqslant q2 \Rightarrow \psi q1 \geqslant \psi q2$
- $p \leqslant \psi\varphi p$ *e* $q \leqslant \varphi\psi q$

Nesse caso, para a conexão de Galois entre P e Q, o mapeamento $\psi\varphi$ é um operador de fecho de P e o mapeamento $\varphi\psi$ é um operador de fecho de Q [Carpineto e Romano, 2004a].

Como será observado nas próximas subseções, conceitos formais da AFC são construídos através de funções $(')$ e $('')$, chamadas operadores de derivação, que formam uma conexão de Galois.

2.3 Análise Formal de Conceitos

A análise formal de conceitos é baseada em três conceitos fundamentais: *contextos formais*, *conceitos formais* e *reticulados conceituais*, que são apresentados nas próximas subseções. O contexto formal (Tabela 1.1) e o reticulado conceitual (Figura 1.1) apresentados na introdução são utilizados para exemplificar esses conceitos.

2.3.1 Contextos Formais

Um contexto formal consiste de dois conjuntos G e M e uma relação de incidência I entre G e M[6]. Um contexto formal tem a notação (G,M,I), sendo $I \subseteq G \times M$ uma relação binária chamada incidência. Os elementos do conjunto G (linhas) são chamados de objetos, enquanto os elementos do conjunto M (colunas) são chamados de atributos. Todo elemento da incidência tem a notação *gIm* ou *(g,m)* $\in I$, que denota um relacionamento entre um objeto e um atributo, que deve ser lido como "o objeto *g* possui o atributo *m*". Observe que um contexto formal representa o universo de estudo, em outras palavras ele é a base de dados que se possui/sabe de um domínio, a partir do qual pretende-se extrair informação e gerar conhecimento.

Dado um conjunto $A \subseteq G$ de objetos de um contexto formal (G,M,I), pode-se pedir quais os atributos de M são comuns a todos os objetos de A. Similarmente, pode-se

[6] As letras G e M possuem origem no Alemão *Gegenstände* e *Merkmale* respectivamente, que significam objetos e características.

pedir, para um conjunto $B \subseteq M$ quais são os objetos que possuem os atributos de B. Essas questões são respondidas pelos operadores de derivação, assim definidos:

$$A' = \{\ m \in M \mid gIm \ \ g \in A\ \} \tag{2.3}$$

$$B' = \{\ g \in G \mid gIm \ \ m \in B\ \} \tag{2.4}$$

Um caso especial dos operadores de derivação ocorre quando os conjuntos de objetos e atributos são vazios; nesse caso, tem-se:

$$A = \oslash \Rightarrow A' = M \qquad B = \oslash \Rightarrow B' = G \tag{2.5}$$

Um exemplo de aplicação da Eq. 2.4 é a derivação do conjunto $B = \{a\}$, em que a representa o atributo *"aquático"* no contexto da Tabela 1.1. Nesse caso, B' produz o conjunto $A = \{2,6\}$, em que 2 representa o atributo *"sapo"* e 6 o atributo *"tubarão"*.

O operador de derivação e suas composições formam importantes propriedades, enunciadas a seguir:

Propriedades 1. *Sejam $A, A_1, A_2 \subseteq G$. Os operadores de derivação satisfazem as seguintes propriedades [Ganter e Stumme, 2003]:*

- $A \subseteq A''$;
- $A_1 \subseteq A_2 \rightarrow A_2' \subseteq A_1'$;
- $A' = A'''$.

Em relação ao conjunto de atributos ($B, B_1, B_2 \subseteq M$) tem-se:

- $B \subseteq B''$;
- $B_1 \subseteq B_2 \rightarrow B_2' \subseteq B_1'$;
- $B' = B'''$.

Essas propriedades são utilizadas por alguns algoritmos para gerar os conceitos formais, construir a relação de cobertura (capítulo 3) e em alguns casos extrair implicações entre atributos [Carpineto e Romano, 2004a].

2.3.1.1 Contextos Formais Multivalorados

Quando um atributo apresentar vários valores a definição vista de contexto formal não é adequada. Nesse caso, os atributos são chamados atributos multivalorados. E contextos formais em que o conjunto de atributos é composto por atributos multivalorados são chamados contextos formais multivalorados. A notação para esse tipo de contexto formal é uma quádrupla *(G, M, V, I)* onde G é um conjunto de objetos, M é um conjunto de atributos multivalorados, V é o conjunto de valores possíveis para os atributos e I é uma relação ternária entre G, M e V ($I \subseteq G \text{x} M \text{x} V$) [Ganter e Wille, 1999].

Em geral, contextos multivalorados podem ser transformados em contextos simples, para se obter os conceitos formais. Uma maneira de fazer isso é criar um novo atributo para cada valor possível de um atributo multivalorado, como no exemplo apresentado na Tabela 2.1; nessa tabela, T_i^j representa o par atributo i, valor j. Note que, o número w_i do par atributo/valor, para cada atributo multivalorado, é a cardinalidade do conjunto V.

É importante ressaltar que os atributos podem ter valores contínuos; por exemplo, V pode ser igual ao conjunto dos números reais $\Re$. Quando isso ocorrer, existirão infinitos pares atributos/valores, pois a cardinalidade do conjunto $\Re$ é infinita. Nesse caso, é interessante escolher intervalos de valores para os atributos. Observe também que, neste caso, $\{T_i^1, \ldots, T_i^{wi}\}$ deve formar uma partição de V.

Tabela 2.1: Obtendo um contexto univalorado a partir de um contexto multivalorado.

Objetos	Atrib nº 1			...	Atrib nº n		
	T_1^1	...	T_1^{w1}	...	T_n^n	...	T_n^{wn}
1	×			...			×
2			×	...	×		
...				...			

2.3.2 Conceitos Formais

Os conceitos podem ser filosoficamente entendidos como unidades básicas do pensamento formados em processos dinâmicos dentro de ambientes sociais e culturais[Wille, 2005]. Em filosofia um conceito é, às vezes, visto como constituído pela sua exten-

são, compreendendo todos os objetos que pertencem ao conceito, e a sua intenção que compreende todos os atributos.

Na AFC, conceitos formais são pares ordenados *(A, B)*, em que A e B são subconjuntos do conjunto de objetos e atributos, ou seja, $A \subseteq G$ (extensão) e $B \subseteq M$ (intenção). Cada objeto da extensão possui todos os atributos da intenção e cada atributo da intenção é um atributo de todos os objetos da extensão. O par *(A, B)* é um conceito formal se, somente se, $A = B'$ e $B = A'$. O conjunto de todos os conceitos formais existentes em um contexto formal recebe a notação $B(G,M,I)$[7].

Um conceito formal pode ser analisado sob outras perspectivas, além do par ordenado (A,B). Na teoria dos grafos um conceito formal representa um máximo subgrafo completo de um grafo bipartite que é descrito pelo contexto formal. Já em uma abordagem mais matemática a geração de conceitos formais pode ser transformada no problema de descobrir conjuntos fechados de uma conexão de Galois.

2.3.3 Reticulados Conceituais

Quando o conjunto de todos os conceitos formais, de um contexto formal (G,M,I), são ordenados hierarquicamente pela ordem da inclusão $\subseteq$ [Davey e Priestley, 1990], ele é chamado *reticulado conceitual*. Como os conceitos formais são obtidos dos operadores de derivação, que formam uma conexão de Galois, os reticulados conceituais também são referenciados na literatura como reticulados de Galois.

Os conceitos formais em um reticulado conceitual são tais que $(A_1,B_1) \leqslant (A_2,B_2)$ quando $A_1 \subseteq A_2$ ($B_2 \subseteq B_1$). Neste caso, o par (A_1,B_1) será chamado subconceito e o par (A_2,B_2) super conceito, se diferentes. A ordem $\leqslant$ é chamada ordem hierárquica ou simplesmente ordem. Para o reticulado mostrado na Figura 1.1, o conceito formal ({*humano*}, {*respiração pulmonar, terrestre, glândulas mamárias, pêlos, raciocínio*}) é, por exemplo, sub-conceito do conceito formal ({*humano, macaco*}, {*respiração pulmonar, terrestre, glândulas mamárias, pêlos*}) com atributo "*raciocínio*" os distinguindo.

Em um reticulado conceitual o ínfimo e o supremo podem ser obtidos pelo seguinte teorema[8]:

[7] A letra *B* possui origem no Alemão *Begriff*, que pode ser entendido como conceito, noção ou percepção.

[8] A prova do teorema pode ser obtida em Davey e Priestley [1990]; Ganter e Wille [1998].

Teorema 1. Seja um contexto formal (G,M,I). Então o reticulado conceitual $B(G,M,I)$ é um reticulado completo no qual, para todo conjunto de conceitos $C \subseteq B(G,M,I)$, o ínfimo e o supremo são descritos por:

$\bigwedge C = (\bigcap X, (\bigcup Y)'')$;

$\bigvee C = ((\bigcup X)'', \bigcap Y)$

sendo que, $X = \{A|(A,B) \in C\}$ e $Y = \{B|(A,B) \in C\}$.

O Teorema 1 é conhecido na literatura como teorema fundamental da análise formal de conceitos.

Como apresentado na Figura 1.1, o reticulado conceitual pode ser representado por um diagrama de linhas. Outras estruturas, como diagrama de linhas aninhado e árvores, também são utilizadas, entretanto, menos frequentemente.

Ao representar uma hierarquia conceitual através de um diagrama de linhas, convenciona-se etiquetar os objetos e atributos apenas uma vez. Desta forma, a análise visual do diagrama torna-se mais fácil. A seguir, os passos fundamentais para representação de uma hierarquia conceitual através de um diagrama de linhas são descritos:

1. para cada conceito formal crie um vértice e posicione-o mais alto que seus subconceitos;
2. crie um relacionamento, através de uma aresta, entre cada vértice e seus vizinhos inferiores;
3. para cada atributo m coloque seu nome no conceito formal $(\{m\}', \{m\}'')$;
4. para cada objeto g coloque seu nome no conceito formal $(\{g\}'', \{g\}')$.

Como será observado no Capítulo 3, essa representação através do diagrama de linhas procura eliminar informação redundante da estrutura do reticulado. Além disso, ela facilita a análise visual do diagrama.

Capítulo 3

Algoritmos

Neste capítulo as principais abordagens e algoritmos para geração de conceitos formais e reticulados conceituais encontradas na literatura são discutidas. Os algoritmos principais são apresentados, em detalhes, um por subseção. Antes são feitas algumas considerações sobre suas complexidades. Em seguida são apresentadas abordagens alternativas para geração dos conceitos formais, apresentado as principais comparações entre algoritmos observadas na literatura e por último são apresentadas abordagens alternativas para redução da complexidade do reticulado conceitual, baseadas no contexto formal e no reticulado conceitual.

3.1 Complexidade do Reticulado Conceitual

A complexidade de um reticulado conceitual pode ser representada como uma função do número de conceitos formais. Entretanto, determinar o número de conceitos formais de um contexto formal (G,M,I) é um problema #P-completo segundo Kuznetsov [2001], fato que posteriormente foi citado e confirmado por Ben-Khalifa e Motameny [2007].

No pior caso o número de conceitos formais possui um comportamento exponencial. Para um contexto formal (G,M,I) esse limite (pior caso) pode ser expresso por $2^{|G|}$ ou $2^{|M|}$, que ocorre em um contexto com $|G| = |M|$ e uma relação de incidência I na qual apenas a diagonal principal da matriz não é preenchida (Tabela 3.1). Observe que esse número é a cardinalidade do conjunto potência de G ou M.

Para contextos formais que apresentam um número de conceitos formais abaixo

Tabela 3.1: Exemplo de contexto formal para o pior caso.

Objetos	**Atributos**			
	Atrib 1	Atrib 2	Atrib 3	Atrib 4
Obj 1		×	×	×
Obj 2	×		×	×
Obj 3	×	×		×
Obj 4	×	×	×	

do limite superior diversos modelos matemáticos são observados na literatura. Esses modelos procuram contornar o fato do problema ser #P-completo e fornecer um valor aproximado do número de conceitos formais.

Para um contexto formal multivalorado (G,M,V,I), em que V é o conjunto de valores possíveis para os atributos, o limite superior do número de conceitos formais é $(|V|+1)^{|M|}$. Outra possibilidade muito interessante, independente do tipo de contexto formal, é expressar esse limite em função do número de incidências $|I|$. Neste caso, Carpineto e Romano [2004a], citando Schutt [1998], apresentam um limite superior de $\frac{3}{2} \times 2^{\sqrt{\{|I|+1\}}} - 1$. Esse limite também foi citado por Kuznetsov [2001] como uma das melhores aproximações existentes.

Observe que a escolha de qual modelo utilizar é uma questão difícil. Entretanto, algumas características do contexto formal e da aplicação podem auxiliar nessa escolha. Se o contexto formal possui uma quantidade maior de objetos, como frequentemente é observado, ou a aplicação/algoritmo é mais sensível a esse parâmetro, um modelo que considere a relevância do número de objetos $|G|$ é mais indicado. Por outro lado, se para a aplicação o número de atributos é mais relevante deve-se escolher o modelo que utilize esse parâmetro.

Contudo, note que o modelo em função do número de incidências I apresenta melhores resultados, conforme citado, por representar a informação contida nos objetos e atributos. Além disso, esse parâmetro possui a informação da correlação entre esses parâmetros. Em outras palavras, o parâmetro I possui uma maior representatividade do contexto formal.

Note que, apesar de existir um limite superior exponencial para alguns domínios específicos, esse comportamento é raramente encontrado na prática. Godin et al. [1986] apresenta um modelo que descreve o comportamento do número de conceitos formais

para uma aplicação de recuperação de informação (RI). Os autores destacam que o comportamento não chega a ser exponencial, pois cada novo objeto possui geralmente no máximo $|M| - 1$ atributos. Note, por exemplo, que, no domínio de recuperação de informação se um atributo representa um termo/palavra e os objetos representam documentos, não é relevante para o modelo um termo comum a todos os documentos.

3.2 O Processo de Construção do Reticulado Conceitual

A construção do reticulado conceitual normalmente consiste de duas fases distintas. Na primeira todos os conceitos formais de um contexto são gerados, e na segunda fase esses conceitos são ordenados hierarquicamente. Na literatura também são encontradas abordagens que combinam ambas as fases em um único processo, como será mostrado.

Em relação aos algoritmos observam-se duas variantes: algoritmos *batch* que consideram o contexto completo (contexto formal estático), no qual o principal representante é o algoritmo *Next Closure* [Ganter, 1984], e algoritmos *incrementais*, que consideram a inserção de atributos e objetos incrementalmente (o contexto formal é dinâmico), no qual um representante é o algoritmo de Godin et al. [1991], que é capaz de tratar a inserção de objetos.

Além disso, observam-se também algoritmos que diferem em relação a certas características. Por exemplo, algoritmos descendentes, que partem do supremo em direção ao ínfimo, ascendentes, que partem do ínfimo em direção ao supremo, algoritmos que enumeram os objetos ou atributos, algoritmos que aplicam o paradigma de divisão e conquista, algoritmos que fazem uso de funções hash, etc.

Nas próximas seções serão apresentados os principais algoritmos disponíveis na literatura capazes de gerar os conceitos formais (Seção 3.3), algoritmos para construir o reticulado conceitual a partir dos conceitos formais (Seção 3.4) e algoritmos para gerar os conceitos e construir o reticulado ao mesmo tempo (Seção 3.5).

3.3 Gerando Conceitos Formais

Antes de uma apresentação dos algoritmos existentes na literatura, pode-se propor um simples algoritmo, porém ineficiente, capaz de gerar os conceitos formais. Conforme apresentado na definição de conceito formal (Subseção 2.3.2) um subconjunto A de G é a extensão de um conceito se, e somente se, $A'' = A$. Note que o dual para o conjunto de atributos também é verdade: um subconjunto B de M é a intenção de um conceito se, somente, se $B'' = B$.

Partindo dessa definição de conceito formal, um algoritmo simples será gerar (A'', A') para todo $A \subseteq G$ ou (B', B'') para todo $B \subseteq M$. Esses subconjuntos podem ser gerados em uma ordem pré-definida. Por exemplo, considere o conjunto de atributos M com n elementos e a existência de uma ordem total imposta aos elementos de M de tal forma que $M = \{m_1 < m_2 < ... < m_n\}$. Note que a ordem utilizada em sua essência não é importante, pois ela é necessária apenas para impor uma ordem que permita a comparação entre todos os elementos.

Partindo da ordem total imposta aos elementos de M pode-se criar uma ordem para os subconjuntos de M, chamada ordem lexicográfica, definida da seguinte forma:

Definição 8. *Sejam $X, Y \subseteq M$. Nesse caso $X < Y$ se, somente, se $\exists i\, m_i \in Y - X$ e $\forall j > i$ $(m_i \in X \Leftrightarrow m_j \in Y)$.*

Em outras palavras, para dois conjuntos X e Y, X será menor que Y, se o maior elemento capaz de diferir os dois conjuntos pertencer a Y.

O procedimento descrito para gerar os subconjuntos do conjunto de atributos M em ordem lexicográfica é aplicado na função *Next Set*. Essa função recebe um conjunto $B \subseteq M$ e retorna o próximo conjunto segundo a ordem lexicográfica. Caso $B = M$ o procedimento retorna $\oslash$.

Para gerar todos os conceitos formais o algoritmo *List of Concepts* (Algoritmo 2) pode ser utilizado. O algoritmo recebe um contexto formal (G, M, I) e retorna um lista C com todos os conceitos formais desse contexto. Para isso, o algoritmo utiliza a função *Next Set* (linha 4) e para cada subconjunto gerado um novo conceito formal é criado e adicionado a lista C de conceitos formais na 5ª linha.

Observe que o procedimento descrito pelo algoritmo *List of Concepts* é extremamente ineficiente, já que o número de subconjuntos de um conjunto é exponencial.

Algoritmo 1 Função Next Set

Entrada: Um conjunto $B \subseteq M$
Saída: Conjunto seguinte a B pela ordem lexicográfica.

```
1: i = 0
2: enquanto i < |M| faça
3:    i = i+1
4:    se m_i ∉ B então
5:       retorne B ∪{m_i}
6:    fim se
7:    B = B −{m_i}
8: fim enquanto
9: retorne ⊘
```

Algoritmo 2 Algoritmo List of Concepts

Entrada: Um contexto formal (G, M, I).
Saída: O conjunto C de conceitos formais de (G, M, I).

```
1: B = G′
2: C = { (B′, B)}
3: enquanto B ≠ M faça
4:    B = Next Set(B)
5:    C = C ∪ { (B′, B″)}
6: fim enquanto
7: retorne C
```

Assim, surge a necessidade de estratégias mais eficientes capazes de gerar todos os conceitos formais.

Uma segunda proposta de algoritmo, mais eficiente, para gerar todos os conceitos formais é baseada na observação que a extensão de cada conceito é a interseção de extensões de atributos existentes [Ganter e Wille, 1999]. Partindo dessa observação pode-se gerar todas as interseções possíveis de extensões de atributos e em seguida formar o conceito formal a partir de cada interseção criada. No algoritmo *Intersection Set* (Algoritmo 3) o procedimento descrito é realizado. Na primeira linha um subconjunto inicial é gerado, já na 4ª linha os próximos subconjuntos são obtidos. Observe que é necessário verificar se o subconjunto já foi gerado (linha 4). Carpineto e Romano [2004a] citam que uma estrutura de dados ótima para esse procedimento é uma árvore *trie*[1], ou árvore de prefixos, já que uma quantidade menor de espaço é necessário para armazenar

[1] O termo *trie* tem origem no inglês "re**trie**val" que pode-ser compreendido como recuperação.

os subconjuntos e o processo de busca de um subconjunto pode-ser realizado de forma eficiente. Por último, na 9ª linha, os conceitos formais são obtidos.

Algoritmo 3 Algoritmo Intersection Set

Entrada: Um contexto formal (G, M, I).
Saída: O conjunto C de conceitos formais de (G, M, I).

```
1:  objs = {G''}
2:  para i = 1 até |M| faça
3:      para cada E ∈ objs faça
4:          objs = objs ∪ { E ∩ {m_i}' }
5:      fim para
6:  fim para
7:  C = ∅
8:  para cada E ∈ objs faça
9:      C = C ∪ {(E, E')}
10: fim para
11: retorne C
```

Note que para gerar todos os conjuntos o algoritmo *Intersection Set* possui um custo de $|M||objs|$, que é equivalente a $|M||C|$. Já para cada atributo m_i é necessário aplicar o operador de derivação ($'$) resultando assim, no pior caso, em um custo de $O(|M||G||C|)$.

Historicamente a primeira proposta de algoritmo para gerar conceitos formais data de 1962[2] com o trabalho de Malgrange [1962], segundo Guénoche [1990]. Entretanto, o primeiro algoritmo prático e citado na literatura é o algoritmo de Chein [1969], que é uma melhoria do trabalho apresentado por Malgrange [1962].

O algoritmo de Chein gera os conceitos formais em um processo *bottom-up*, analisando o contexto formal (matriz) em camadas, em que a camada L_n representa todos os conceitos formais cuja intenção possui cardinalidade n. O algoritmo representa os conceitos pelo par extensão e intenção e gera a extensão de cada novo conceito como a união de duas extensões de conceitos já existentes. Partindo desse princípio, a complexidade do algoritmo segundo Kuznetsov e Obiedkov [2002] é $O(|G|^3|M||L|)$, em que L representa o número de camadas. O objetivo do algoritmo é gerar submatrizes máximas de uma matriz global, o que é equivalente a gerar conceitos formais.

[2]O leitor mais atento irá constatar que alguns algoritmos são datados anteriormente ao surgimento da análise formal de conceitos, que foi proposta em 1982 com o trabalho apresentado por Wille [1982]. Esses algoritmos posteriormente foram adaptados ou aplicados sob essa nova perspectiva.

Outro algoritmo capaz de gerar os conceitos formais é o algoritmo de Fay [1973], que é pouco referenciado/utilizado na literatura. O primeiro trabalho eficiente encontrado na literatura é o algoritmo de Norris [1978]. Ele também utiliza a ideia de camadas para encontrar os retângulos máximos do contexto (conceitos formais) e apresenta uma complexidade de $O(|G|^2|M||L|)$.

3.3.1 Algoritmo Next Closure

De todos os algoritmos observados na literatura capazes de apenas gerar os conceitos formais, o algoritmo *Next Closure* é o mais relevante. Como comentado pode-se gerar o conjunto de todos os conceitos formais de um contexto (G,M,I) a partir dos elementos do conjunto potência de M. Contudo, esse procedimento apresenta um comportamento exponencial; além disso, para cada conjunto obtido é necessário verificar se ele já foi gerado. Para evitar esses problemas, pode-se gerar apenas os conjuntos fechados em uma ordem pré-definida.

O algoritmo *Next Closure*[3] foi proposto em 1984 por Ganter [1984] com o objetivo de encontrar conjuntos fechados, que é equivalente a obter conceitos formais. O algoritmo é baseado na existência de uma ordem total para os atributos M do contexto formal, uma ordem lexicográfica para os subconjuntos de M (Definição 8) e na utilização do operador de derivação ($''$).

Assim, pode-se enunciar um pseudocódigo do algoritmo *Next Closure* (Algoritmo 4). O algoritmo parte do princípio que para qualquer conjunto de objetos B, se $B'' - B$ contém apenas elementos menores que os elementos contidos no conjunto B, então todos os subconjuntos contidos entre B e B'' serão iguais a B'' ou serão gerados posteriormente[Carpineto e Romano, 2004a]. Desta forma é possível não examinar todos os subconjuntos do conjunto de atributos M, o que proporciona uma grande vantagem ao algoritmo. Além disso, observe que cada conjunto fechado é gerado uma única vez, o que elimina a necessidade de uma estrutura de busca auxiliar.

Já para gerar todos os conceitos formais o algoritmo *All Concepts* (Algoritmo 5) pode ser utilizado. Em síntese, o algoritmo realiza os seguintes procedimentos: para cada conjunto fechado, começando com G', ele gera o conjunto fechado B (linha 4) e forma um novo conceito formal (B',B), que é adicionado a lista de conceitos formais C

[3]Na literatura o algoritmo também é referenciado pelo nome do autor, isto é, algoritmo de Ganter.

Algoritmo 4 Função Next Closure

Entrada: Um contexto formal (G,M,I) e $B \subseteq M$.
Saída: O próximo conjunto fechado maior que B.

```
1:  i = 0
2:  enquanto i<|M| faça
3:     i = i+1
4:     se  m_i ∉ B então
5:        B = B ∪{m_i}
6:        se  ∄ j>i m_j ∈ B'' - B então
7:           retorne  B''
8:        fim se
9:     fim se
10:    B = B - {m_i}
11: fim enquanto
12: retorne  B
```

(linha 5).

Algoritmo 5 Algoritmo All Concepts

Entrada: Um contexto formal (G,M,I).
Saída: O conjunto C de conceitos formais de (G,M,I).

```
1: C = {(G'', G')}
2: B = G'
3: enquanto B ≠ M faça
4:    B = Next Closure((G,M,I), B)
5:    C = C ∪ {(B', B)}
6: fim enquanto
7: retorne  C
```

A capacidade do algoritmo de saltar a avaliação de alguns subconjuntos é muito interessante. Entretanto, sua complexidade no pior caso continua exponencial, isto é, $O(|C||G||M|)$, em que O(|G||M|) é o tempo de computação dos operadores de derivação $(')$ e $('')$ e $|C|$ é o número de conceitos formais, que no pior caso apresenta um comportamento exponencial $2^{|M|}$.

Conforme poderá ser observado na Seção 3.7, que trata a comparação entre os algoritmos, o algoritmo *Next Closure* é o mais indicado para bases de dados reais quando se deseja apenas gerar os conceitos formais. Isso se deve à capacidade do algoritmo de realizar saltos durante a busca/geração dos conceitos e a geração de cada conceito uma

única vez, o que evita a necessidade da realização de buscas, reduzindo assim tempo e espaço. Por outro lado, quando é necessário gerar o reticulado conceitual nem sempre o algoritmo *Next Closure* é indicado, pois existe um custo adicional para a posterior ordenação dos conceitos no reticulado. Observe que essa dificuldade, gerar o reticulado conceitual juntamente com os conceitos, deve-se à não existência de uma relação entre a ordem lexicográfica utilizada e a relação de cobertura dos conceitos.

3.4 Obtendo o Reticulado Conceitual a partir dos Conceitos Formais

Conforme apresentado na Seção 3.3, vários algoritmos são capazes de gerar apenas os conceitos formais. Apesar de existir aplicações para esses algoritmos, em muitos casos também é necessário gerar o reticulado conceitual.

A partir de uma lista C de conceitos formais, pode-se construir o reticulado em um processo semelhante a uma ordenação. Um procedimento possível é percorrer a lista de conceitos C com duas repetições aninhadas e comparar os conceitos indexados em cada repetição. Em Godin et al. [1995], alguns algoritmos são propostos para encontrar a relação de cobertura dos conceitos; em sua essência os algoritmos aplicam esse mesmo processo. Mais recentemente, Martin e Eklund [2008] apresentam um algoritmo para geração do reticulado conceitual a partir dos conceitos formais chamado *Intents Only*. O algoritmo possui a vantagem de não necessitar do contexto formal (G,M,I) para encontrar a relação de cobertura entre os conceitos, em outras palavras ele não necessita aplicar os operadores de derivação.

3.4.1 Algoritmo Concepts Cover

Um algoritmo mais conhecido e capaz de gerar a relação de cobertura a partir de um lista C de conceitos formais é o algoritmo *Concepts Cover* (Algoritmo 6) proposto por Nourine e Raynaud [1999]. Em síntese o algoritmo realiza o seguinte procedimento: cada conceito $c \in C$ é analisado e os conceitos formais inferiores são identificados através do cálculo de interseções (linha 5) em um processo semelhante ao aplicado pelo algoritmo *Intersection set* (Algoritmo 3). Observe que um problema do procedimento descrito é a necessidade da identificação dos conceitos formais que são filhos do con-

ceito atual, já que todos os conceitos descendentes são encontrados. Para facilitar isto, contadores auxiliares são utilizados. O contador de cada conceito é incrementado sempre que um conceito candidato a ser filho é recuperado, contudo ele será um filho do conceito somente se seu contador for igual à diferença entre a intenção do conceito formal candidato e a intenção do conceito pai (linha 7). Note que *contador* é um vetor indexado pelos conceitos formais. Esse procedimento garante que apenas os filhos do conceito em análise sejam selecionados[Carpineto e Romano, 2004a].

Algoritmo 6 Algoritmo Concepts Cover

Entrada: O conjunto C de conceitos formais e um contexto formal (G,M,I).
Saída: Um reticulado conceitual $L = B(G,M,I)$.

```
1:  para cada (X,Y) ∈ C faça
2:     ∀ c ∈ C faça contador[c] = 0
3:     para cada m ∈ M - Y faça
4:        inters = X ∩ m'
5:        encontre (X1,Y1) ∈ C tal que X1 = inters
6:        contador[(X1,Y1)] = contador[(X1,Y1)] + 1
7:        se ( |Y1| - |Y| ) = contador[(X1,Y1)] então
8:           indique que (X1,Y1) ≺ (X,Y) em L
9:        fim se
10:    fim para
11: fim para
12: retorne L
```

A construção do reticulado a partir do conjunto de conceitos pode parecer ineficiente; entretanto, deve-se lembrar que pode-se tirar vantagem do fato que todos os conceitos já foram previamente gerados. Logo, desde que seja possível realizar uma busca no conjunto de conceitos de forma eficiente, a construção do reticulado a partir do conjunto de conceitos pode ser bem menos custoso que a execução de um algoritmo de construção do reticulado a partir do contexto formal. Deve-se, portanto, analisar qual estratégia é mais adequada. A complexidade do algoritmo *Concepts Cover* segundo Carpineto e Romano [2004a] é $O(|C||M|(|G|+|M|))$.

3.5 Obtendo o Reticulado Conceitual e os Conceitos Formais

Como supracitado, na maioria das aplicações é necessário gerar o reticulado conceitual; assim, pode-se tirar proveito da relação de cobertura e gerar o reticulado na medida em que os conceitos formais são obtidos.

O primeiro algoritmo capaz de gerar os conceitos formais e o reticulado conceitual em um único processo é o algoritmo *Bordat* [Bordat, 1986]. Ele utiliza uma estratégia *top-down*, que inicia a geração dos conceitos a partir do supremo. Assim, a relação hierárquica é claramente definida para todos os conceitos.

Seguindo a ordem cronológica tem-se os algoritmos de Godin et al. [1991] e Carpineto e Romano [1993], que são algoritmos incrementais descritos na Subseção 3.5.2.

Note que, existem outros algoritmos, como *Dowling, Close by One, Lindidg, Nourine*[Kuznetsov e Obiedkov, 2001, 2002] e *extensões do algoritmo Bordat*[Bertet et al., 2007]. Contudo, a grande maioria são versões dos algoritmos anteriores, sendo em geral pouco referenciados.

3.5.1 Algoritmo Next Neighbours

Um dos melhores e mais simples algoritmos conhecidos na literatura para construção do conjunto de conceitos formais e sua ordenação em um diagrama de linhas é o algoritmo *Next Neighbours* [Carpineto e Romano, 2004a], que é baseado na relação de cobertura dos conceitos formais. O algoritmo pode ser considerado uma evolução do algoritmo *Bordat*.

Iniciando do conceito supremo, que por definição é o conceito formal (G, G'), o algoritmo constrói um nível por vez, onde o próximo nível contém os filhos dos conceitos do nível atual, em claro processo *top-down* (Algoritmo 7).

Para calcular os filhos de um conceito formal a função *Find Lower Neighbours* (Algoritmo 8) é utilizada. Ela é baseada na observação que os vizinhos inferiores de um conceito (X,Y) são formados pela adição de alguns novos atributos.

Segundo Carpineto e Romano [2004a], o algoritmo terá um custo de $O(|C||G||M|^2)$, onde C representa a estrutura que armazena os conceitos já gerados. Uma versão dual do algoritmo pode ser utilizada construindo-se o reticulado do ínfimo para supremo, isto

Algoritmo 7 Algoritmo Next Neighbours

Entrada: Um contexto formal (G,M,I).
Saída: Um reticulado conceitual $L = B(G,M,I)$.

```
1:  C = {(G,G′)}
2:  L = ⊘
3:  currentLevel = {(G,G′)} //nível inicial
4:  enquanto currentLevel ≠ ⊘ faça
5:     nextLevel = ⊘
6:     para cada (X,Y) ∈ currentLevel faça
7:        lowerNeighbours = FindLowerNeigbours( (X,Y) )
8:        para cada (X1,Y1) ∈ lowerNeighbours faça
9:           se (X1,Y1) ∉ C então
10:             C = C ∪ {(X1,Y1)}
11:             nextLevel = nextLevel ∪ {(X1,Y1)}
12:          fim se
13:          indique que (X1,Y1) ≺ (X,Y) em L
14:       fim para
15:    fim para
16:    currentLevel = nextLevel
17: fim enquanto
18: retorne L
```

Algoritmo 8 Função Find Lower Neighbours

Entrada: Um conceito formal (X,Y).
Saída: O conjunto de conceitos formais filhos de (X,Y).

```
1: filhos = ⊘
2: para cada m ∈ M-Y faça
3:    X1 = (Y ∪ {m})′
4:    Y1 = X1′
5:    se (X1,Y1) é filho de (X,Y) então
6:       filhos = filhos ∪ {(X1,Y1)}
7:    fim se
8: fim para
9: retorne filhos
```

é da base (M', M) para o topo (G, G'); neste caso, tem-se um custo $O(|C||G|^2|M|)$.

3.5.2 Algoritmos Incrementais

Em muitas aplicações é necessária a atualização do contexto formal e, consequentemente, do reticulado conceitual, que pode ocorrer com a inserção ou exclusão de objetos ou atributos; dentre elas, a inserção de objetos é mais comum.

O primeiro algoritmo com essa capacidade, tratar um contexto formal dinâmico, foi proposto por Godin em 1991 (Subseção 3.5.2.1)[4]. Posteriormente, observam-se os algoritmos de Carpineto e Romano [1993] e Qiao et al. [2003], que são adaptações do algoritmo de Nourine capaz de atualizar o reticulado. Já em Valtchev e Missaoui [2002] é apresentada uma proposta de construção do reticulado conceitual baseada no paradigma de divisão e conquista. A partir da divisão do contexto formal gera-se os reticulados individualmente e, em seguida, a junção desses reticulados é realizada, obtendo-se assim um reticulado conceitual global. A vantagem da técnica, segundo os autores, é permitir a atualização do reticulado com a inserção de um conjunto de objetos ao mesmo tempo.

3.5.2.1 Algoritmo Godin

Em seu trabalho, Godin et al. [1991] destacam a necessidade de algoritmos capazes de gerar os conceitos formais e construir o reticulado conceitual ao mesmo tempo. Dos algoritmos existentes até então na literatura, apenas o algoritmo *Bordat* era capaz de realizar os dois processos ao mesmo tempo. Godin et al. [1991] também destacam que, em aplicações reais, é necessária a atualização do reticulado conceitual quando novos objetos são inseridos no contexto formal; e que, até então, não eram encontrados algoritmos com essa capacidade na literatura. A partir dessas constatações os autores propõem um novo algoritmo capaz de construir o reticulado conceitual gradativamente através da inserção de novos objetos.

Um possível algoritmo consiste em atualizar os conceitos que sofrem alguma alteração com a inserção de um novo objeto e a criação de novos conceitos, caso seja necessário, conforme procedimento realizado pelo algoritmo *Godin* (Algoritmo 9). Para um objeto h, com os atributos h', os novos conceitos são sempre da forma $(X \cup \{h\}, Y \cap h')$

[4]Na literatura o algoritmo *Godin* também é conhecido por *algoritmo de atualização pela cardinalidade*, *update by cardinality algorithm* em inglês.

para algum conceito $(X,Y) \in L$ (linha 16). Assim, o algoritmo baseia-se no fato da extensão de um conceito formal ser formada pelo acréscimo do objeto h à extensão anterior e da intenção ser formada pela interseção da intenção anterior com os atributos h'. Observe que para evitar a necessidade de analisar todos os conceitos formais eles são agrupados pela cardinalidade da intenção (linha 1).

Além da atualização dos conceitos formais é necessário a atualizar as arestas. Quando um novo conceito formal é criado esse deve ser conectado ao reticulado, sendo ligado a conceitos já existentes, outro novo conceito ou ambos. Já quando um conceito é apenas atualizado seu pai não é atualizado. Entretanto, os filhos desse conceito atualizado podem sofrer alterações.

As principais vantagens da utilização do algoritmo *Godin* são a não utilização dos operadores de derivação e a facilidade para detectar se um novo conceito já existe no reticulado, que pode ser feita através de busca nos grupos com a mesma cardinalidade. Cabe ressaltar que o algoritmo *Godin* aqui apresentado, foi baseado no trabalho de Carpineto e Romano [2004a]; ambas as versões são idênticas, porém a última foi apresentada de uma forma mais didática.

Em relação à complexidade do algoritmo proposto, Godin et al. [1991] afirmam que apesar da complexidade do reticulado ser exponencial no número de conceitos, na prática esse comportamento não é observado. Desta forma, ainda segundo os autores, a complexidade do algoritmo proposto será de $O(|G|2^K)$, em que K é o número de atributos do objeto que possui a maior quantidade de atributos.

Note que o algoritmo *Godin* descrito trata apenas da inserção de objetos no reticulado conceitual. Caso sejam necessárias outras atualizações no contexto, uma boa referência é o trabalho mais atual de Carpineto e Romano [2004a], o qual descreve um algoritmo para cada atualização possível (inserção ou deleção de atributos e inserção ou deleção de objetos).

3.6 Abordagens Alternativas para Geração dos Conceitos Formais

Como o custo computacional da análise formal é proibitivo em muitas aplicações, abordagens alternativas para gerar os conceitos formais e o reticulado conceitual são cons-

Algoritmo 9 Algoritmo Godin

Entrada: Um reticulado conceitual $L = B(G,M,I)$, um novo objeto h e os atributos h'.
Saída: O reticulado conceitual L atualizado.

```
1: para j = 0 até |M| faça
2:    oldBubcket[j] = { (X,Y) ∈ L | |Y| = j }
3:    newBubcket[j] = ⊘
4: fim para
5: para j = 0 até |M| faça
6:    para cada (X,Y) ∈ oldBucket[j] faça
7:       se Y ⊆ h' então
8:          //alterando conceito formal existente
9:          substitua (X,Y) em L por (X ∪ {h}, Y)
10:         newBucket[j] = newBucket[j] ∪ {(X,Y)}
11:         se Y = h' então
12:            retorne L
13:         fim se
14:      senão
15:         inters = Y ∩ h'
16:         se ∄ {(X1,Y1) ∈ newBucket[|inters|] | Y1 = inters} então
17:            //novo conceito formal
18:            nConcept = (X ∪ {h}, inters)
19:            newBucket[|inters|] = newBucket[|inters|] ∪ {nConcept}
20:            //atualizando aresta
21:            indique que (X,Y) ≺ nConcept em L
22:            para k = 0 até (|inters| - 1) faça
23:               para cada (X1,Y1) ∈ newBucket[k] faça
24:                  se Y1 ⊂ inters e ∄ (X2,Y2) filho de (X1,Y1) | Y2 ⊂ inters então
25:                     indique que nConcept ≺ (X1,Y1) em L
26:                     se (X1,Y1) é pai de (X,Y) então
27:                        remova (X,Y) ≺ (X1,Y1) em L
28:                     fim se
29:                  fim se
30:               fim para
31:            fim para
32:         fim se
33:      fim se
34:      se inters = h' então
35:         retorne L
36:      fim se
37:   fim para
38: fim para
39: retorne L
```

tantemente observadas. Em geral, a paralelização dos algoritmos é utilizada. Uma forma simples de gerar os conceitos em paralelo é dividindo o contexto formal e distribuindo as partes para vários nós, cada um responsável por gerar o conceito individualmente e ao final enviá-los para um coordenador central, que por sua vez é responsável por unir os conceitos.

Uma abordagem de paralelização mais prática e associada com outros conceitos é apresentada por Fu e Nguifo [2003]. No trabalho, os quatro principais algoritmos, *Chein, Norris, Next Cloure* e *Bordat*, encontrados na literatura para geração de reticulados conceituais são comparados e, segundo os autores, os demais algoritmos encontrados são em geral variações desses quatro. Os autores também citam que esses quatro algoritmos possuem um foco nos objetos ou nos atributos, e que o desempenho deles pode ser diferente, dependendo da quantidade de objetos e atributos, o que é constantemente observado na literatura.

Com essas observações os autores propõem uma nova definição: um algoritmo dual, o qual é um algoritmo com foco nos objetos e atributos. Usualmente a definição de algoritmo dual é considerada a mesma na literatura, pois os reticulados finais são isomorfos; entretanto, os autores afirmam que frequentemente ambas as versões não apresentam o mesmo tempo computacional. Nos resultados experimentais apresentados essa afirmação é confirmada.

A partir dos experimentos realizados os autores apresentam uma versão paralela do algoritmo *Next Closure*, chamada *Scaling Next Closure*, que se baseia na divisão de um contexto formal (G,M,I) em subespaços de busca. Primeiramente, a proposta necessita de um contexto ordenado, utilizando como comparador o número de objetos de cada atributo m_i. Em segundo lugar, ela usa o conceito de conjunto dobra F [5] de um atributo $m_i \in M$, que consiste de todos os elementos maiores que m_i em M, isto é, $F_{mi} = \{m_j \in M |$ para todo $i < j \leqslant |M|\}$.

Observe que desta forma os autores criaram um espaço de busca para cada atributo. Desta forma, para cada conceito é possível reduzir o espaço de busca do atributo m_i de 2^n para o mínimo de $(2^n, 2^{m-i})$, onde n é o número de objetos do atributo a_i. Assim a abordagem proposta pode ser sumarizada em:

- construa um contexto formal ordenado;

[5] Do Inglês: *fold set*

- gere os espaços de busca;
- aplique o algoritmo *Next Closuse* em cada espaço de busca construído.

A principal vantagem da proposta apresentada é criar uma geração de conceitos formais paralela, onde cada espaço de busca é independente.

3.7 Comparação entre Algoritmos

Em Guénoche [1990] um primeiro estudo comparativo de algoritmos para geração do conjunto de conceitos formais e construção do reticulado conceitual foi realizado. No trabalho quatro algoritmos, *Chein, Next Closure*, *Norris* e *Bordat*, são analisados e as complexidades são apresentadas. Dentre as conclusões destacam-se que o algoritmo *Norris* é "um pouco" superior ao algoritmo *Next Closure*, apesar deste último apresentar uma estratégia melhor para obter os conceitos, e que o algoritmo *Bordat* é mais indicado quando se deseja construir o reticulado conceitual, já que ele é capaz de construir o reticulado enquanto obtém os conceitos formais.

Em Godin et al. [1995] é feita uma comparação de seu algoritmo, previamente apresentado pelos autores em Godin et al. [1991], com os algoritmos *Bordat, Next Closure* e *Chein*. Como os algoritmos *Next Closure* e *Chein* não geram o reticulado conceitual, os autores apresentam adaptações para gerar o reticulado. Os autores citam que a mesma estrutura básica foi utilizada na comparação, fato fundamental em tais comparações; entretanto, não são apresentadas descrições sobre tais estruturas. Já em relação aos testes, estes foram realizados utilizando dados reais e valores aleatórios obtidos variando G, M e K, em que K representa o número médio de atributos de cada objeto. Dentre os resultados apresentados, destaca-se o fato do algoritmo incremental apresentar um desempenho superior em muitos casos mesmo para instâncias pequenas, fato que não era esperado. Em apenas alguns casos o algoritmo *Chein* foi capaz de superar o algoritmo incremental proposto. Já os conhecidos algoritmos *Next closuse* e *Bordat* apresentaram um desempenho muito ruim. Quando a instância do problema aumenta o desempenho do algoritmo incremental torna-se muito superior; segundo os autores a atualização incremental apresenta um comportamento proporcional ao número de instancias previamente tratadas, fato que justifica o comportamento constatado. Como ponto falho do

trabalho nota-se a pouca descrição dos dados utilizados para teste, o tamanho relativamente reduzido das instâncias utilizadas, a falta de comparação dos algoritmos com o pior caso e a utilização de adaptações nos algoritmos não comprovadamente eficientes.

Em Kuznetsov e Obiedkov [2001] e Kuznetsov e Obiedkov [2002] uma comparação teórica (pior caso) e experimental de 10 algoritmos para obtenção dos conceitos formais e construção do reticulado conceitual foi realizada. Os testes foram realizados utilizando contextos aleatórios (variando $|G|$, $|M|$ e o número de atributos por objeto). Como resultado, os autores indicam o algoritmo *Godin* para contextos pequenos e esparsos; para contextos densos os algoritmos *Close by One* ou *Norris* são indicados. Já para contextos com uma densidade média o algoritmo de *Bordat* é o mais aconselhado.

Kuznetsov e Obiedkov [2002] também criticam os primeiros trabalhos de comparação de algoritmos. Segundo os autores o trabalho de Guénoche [1990] não apresenta os algoritmos de uma forma correta e nenhuma informação sobre os dados utilizados para testes é apresentada. De fato, observa-se que o trabalho é um resumo dos algoritmos e que essas informações relevantes não são fornecidas. Já em relação ao trabalho de Godin et al. [1995], eles criticam que apenas algoritmos para construir o reticulado conceitual foram considerados, e aqueles que originalmente apenas geravam os conceitos formais foram alterados. Entretanto, segundo Kuznetsov e Obiedkov [2002], essas alterações podem aumentar o custo dos algoritmos drasticamente.

Já nos testes realizados por Fu e Nguifo [2004], foi observado que o algoritmo *Next Closure* apresenta um desempenho superior. Em bases de dados reais, frequentemente são observados menos atributos que objetos, o que sugere a utilização do algoritmo *Next Closure*. Quando comparado com um contexto no pior caso (contexto com apenas a diagonal principal não preenchida), o algoritmo também se apresentou superior aos demais. Entretanto, o algoritmo necessita de um pós-processamento para construir a relação de cobertura dos conceitos, o que pode provocar uma queda em seu desempenho total.

Quando se comparam algoritmos para geração de conceitos formais ou reticulados, não é adequada a utilização de bases de testes aleatórias ou de instâncias muito pequenas, pois essas podem mascarar o desempenho dos algoritmos analisados em situações reais. Segundo Fu e Nguifo [2004] quando esses algoritmos são aplicados em mineração de dados essa comparação é preocupante e bases de dados reais podem ajudar a entender melhor o comportamento real desses algoritmos. Entretanto, o autor não leva em consi-

deração que a comparação desses algoritmos com bases de dados reais pode desviar o foco principal do trabalho, já que muitas vezes é necessário todo um pré-processamento nessas bases de dados. Assim, mais uma vez fica evidente a necessidade de um conjunto de bases de testes pré-estabelecidas para comparação experimental desses algoritmos.

Fu e Nguifo [2004] também criticam alguns trabalhos prévios de comparação de algoritmos. Para os autores, assim como Kuznetsov e Obiedkov [2002], o trabalho de Guénoche [1990] não apresenta informações importantes sobre os dados utilizados nas comparações. Sobre o trabalho de Godin e Chau [2000], os autores afirmam que as instâncias de testes utilizadas foram muito pequenas e não representavam a realidade. Já sobre o trabalho de Kuznetsov e Obiedkov [2002], é citado que os testes foram realizados apenas com dados aleatórios e pequenos e que apenas uma base de dados real foi considerada.

3.8 Alternativas para Redução da Complexidade

Como supracitado, a complexidade, em função do número de conceitos formais, é um fator determinante para aplicação da análise formal de conceitos. Por esse motivo, métodos para redução da complexidade, baseados em algoritmos, podas, heurísticas, detecção de relevância, etc, são extremamente importantes. Nesta seção, vários métodos para redução da complexidade do reticulado conceitual observados na literatura são apresentados, sendo que aqueles mais relevantes/importantes são apresentados com mais detalhes. Cabe ressaltar que as abordagens apresentadas nesta subseção não tratam de melhorias nos algoritmos existentes.

As abordagens discutidas podem ser classificadas em dois grandes grupos: *baseadas no conhecimento do usuário* e *baseadas na eliminação de informação redundante*. As técnicas baseadas no conhecimento do usuário são aquelas nas quais um ou mais parâmetros são fornecidos. Em geral, esses parâmetros são algum tipo de limiar capaz de distinguir a informação considerada irrelevante pelo usuário. Já as técnicas baseadas na eliminação de informação redundante, procuram eliminar qualquer informação redundante que o reticulado possa estar representando.

Primeiramente, antes da apresentação dessas técnicas, é necessário definir o conceito de informação redundante.

Definição 9. *Seja (G,M,I) um contexto formal e $B(G,M,I)$ um reticulado conceitual construído a partir desse contexto. Uma informação será chamada redundante se após sua eliminação ainda for possível obter (G,M,I) a partir de $B(G,M,I)$, e vice-versa.*

Em outras palavras, se após a eliminação da informação redundante do contexto formal ainda for possível obter um reticulado conceitual isomorfo ao reticulado original e a partir do reticulado conceitual, sem informação redundante, ainda for possível obter o contexto formal original, então a informação eliminada será chamada redundante.

Observe que os objetos G e atributos M do contexto formal podem ser recuperados a partir do supremo e ínfimo do reticulado conceitual, respectivamente. Já a relação de incidência I pode ser recuperada a partir de cada conceito formal de $B(G,M,I)$, ou formalmente $I = \bigcup\{A \times B | (A,B) \in B(G,M,I)\}$.

Nas subseções a seguir as técnicas de redução da complexidade do reticulado são apresentadas.

3.8.1 Redução Baseada no Contexto Formal.

A primeira simplificação possível no contexto formal é a junção de linhas e/ou colunas idênticas [Ganter e Wille, 1999]. Seja $g,h \in G$. Se $g' = h'$ então g e h podem ser representados por um único objeto j. Dualmente, seja $m,n \in M$. Se $m' = n'$ então m e n podem ser representados por um único atributo i. Um contexto formal com essas características é chamado na literatura de *contexto formal limpo*[6].

Ganter e Wille [1999] também citam a existência de um conjunto mínimo de objetos G', atributos M' e incidências I' de um contexto formal (G,M,I) capaz de gerar um reticulado conceitual $B(G',M',I')$ isomorfo ao reticulado conceitual $B(G,M,I)$ original. Wenxiu et al. [2005] estendem esse conceito de conjunto mínimo e propõem uma técnica teórica para redução do número de atributos do reticulado conceitual. Os autores provam que para todo contexto formal existe uma redução. Contudo, nenhuma proposta prática ou algorítmica é apresentada pelos autores.

Essas propostas não alteram a estrutura do reticulado conceitual[7]; entretanto, alguns objetos ou atributos podem não aparecer. Cabe ao usuário determinar se o objeto e/ou

[6] Do Inglês: *formal concept clarified.*

[7] No sentido em que o número de conceitos formais e a relação de incidência das arestas não são alterados.

atributo removido não influenciará na aplicação que faça uso do reticulado conceitual.

Outra possibilidade interessante é aplicar técnicas do processo KDD (*knowledge discovery in databases*)[Kantardzic, 2003] para reduzir o contexto. Gajdos et al. [2004] e Cheung e Vogel [2005] propõem a utilização de SVD (*singular value decomposition*), que é uma técnica capaz de reduzir a dimensionalidade de massas de dados. A proposta dos autores associa/junta atributos, o que acaba alterando a estrutura do reticulado construído. Já Xie Cui-hua e Jian-li [2005] propõem a utilização de um modelo chamado *modelo de nuvens*[8]. O modelo é utilizado para construção de contextos formais em que os atributos originalmente possuem valores contínuos. Ele propicia a escolha de atributos com maior representatividade.

A proposta apresentada por Xie Cui-hua e Jian-li [2005] pode ser comparada com as técnicas de discretização utilizadas no processo KDD[Yang e Webb, 2005], que procuram transformar dados contínuos em valores discretos, localizando pontos de corte ótimos nos atributos e mantendo o máximo possível da qualidade da informação.

Outra técnica para redução do reticulado conceitual baseada no contexto formal, muito interessante, foi apresentada por Belohlávek et al. [2004]. Os autores propõem utilizar o conhecimento do usuário para criar restrições nos atributos do contexto formal. Essas restrições são chamadas AD-formulas (*attribute-dependency formulas*) e são representadas da seguinte forma: $m \sqsubseteq m_1 \sqcup ... \sqcup m_n$. Para o exemplo dos animais (Tabela 1.1) um exemplo de AD-formula seria: *raciocínio* $\sqsubseteq$ *pêlos* $\sqcup$ *glândulas mamárias*. Desta forma, durante a geração dos conceitos formais e construção do reticulado conceitual, apenas conceitos formais com a restrição proposta seriam considerados. Em outras palavras, uma restrição é criada baseada no conhecimento do usuário para selecionar apenas conceitos formais interessantes.

Já Carpineto e Romano [2004a] apresentam um arcabouço para eliminação de informação semelhante a apresentada por Belohlávek et al. [2004]. Contudo, as restrições são aplicadas diretamente no reticulado conceitual e apenas conceitos consistentes, que atendem um conjunto de restrições, são selecionados.

[8]Do Inglês: *cloud model*.

3.8.2 Redução Baseada no Reticulado Conceitual.

Um reticulado conceitual em sua essência apresenta muita informação redundante graças à herança existente/representada na estrutura. Seja (X,Y) um conceito formal. Nesse caso, o conjunto de atributos Y possui todos os atributos dos ancestrais; similarmente, o conjunto de objetos X estará presente em todos os descendentes do conceito formal. A eliminação dessa informação redundante é uma primeira simplificação possível do reticulado conceitual, que pode ser realizada sem perda de informação.

Como demonstrado na Subseção 2.3.3, ao representar um reticulado conceitual através de um diagrama de linhas procura-se eliminar a informação redundante representando cada atributo m e objeto g uma única vez. Baseado nesse conceito de redução da informação do reticulado conceitual, Arévalo et al. [2007] propõe a geração de sub-hierarquias conceituais[9]. Nesse caso são selecionados apenas os conceitos formais $C^g = \{(g'',g')|g \in G\}$, chamados conceitos objetos, e os conceitos formais $C^m = \{(m'',m')|m \in M\}$, chamados conceitos atributos. Nessa proposta o reticulado conceitual é construído apenas com os conceitos formais $C^g \cup C^m$. Esse tipo de redução é muito utilizado para análise visual do reticulado conceitual, por manter a estrutura principal do reticulado.

Já Wang e Zhang [2008] apresentam uma técnica teórica para redução da complexidade do reticulado conceitual que procura selecionar um conjunto mínimo de atributos, capaz de preservar todos os conceitos formais e a hierarquia conceitual. A proposta dos autores baseia-se na junção de atributos. Em outras palavras a proposta é encontrar um conjunto mínimo de atributos, ou um conjunto mínimo de objetos para o dual, que é capaz de produzir um reticulado conceitual isomorfo ao reticulado com todos os atributos. Contudo, não são apresentadas pelos autores informações práticas sobre a proposta. Observe que nesse caso, diferentemente do trabalho apresentado por Wenxiu et al. [2005], a junção dos objetos ou atributos é baseada na estrutura do reticulado conceitual.

Outra opção para reduzir a complexidade do reticulado conceitual é associar o conceito de itens frequentes, provenientes do processo KDD, a conceitos formais frequentes. A partir de uma base de dados D, com n parâmetros e T_m registros ou transações, pode-se perguntar: quais atributos são frequentes na base de dados D? Essa pergunta é respondida através da mineração de itens frequentes [Kantardzic, 2003]. Similarmente,

[9] Do Inglês: *Galois Subhierarchy.*

dado um contexto formal (G,M,I) pode-se perguntar: quais atributos $m \in M$ são frequentes no conjunto de objetos G? Note que, em essência, um contexto formal é uma representação para uma base de dados D na qual todos os parâmetros T são discretizados, representados por M e chamados de atributos. Assim, todo arcabouço matemático da análise formal de conceitos pode ser utilizado para extrair itens frequentes. De fato, na literatura existe uma grande interdisciplinaridade entre o processo KDD e a AFC.

Inicialmente, dois conceitos são necessários, o conceito de *suporte* e o conceito de *conjuntos frequentes*. O suporte de um de conjunto $S \subseteq M$ ($sup(S,G)$) é número de objetos em G que contém S. O conjunto $S \subseteq M$ será dito frequente se, somente se, $sup(S,G) \geqslant minSup$, em que $minSup$ é um suporte mínimo fornecido como parâmetro pelo usuário.

Assim, pode-se agora relacionar itens frequentes e conceitos formais, conforme definido a seguir:

Definição 10. *Um conceito formal (X,Y) será frequente se, somente se, $sup(Y,G) \geqslant minSup$.*

A relação entre conceitos formais frequentes e itens frequentes é possível devido ao suporte de um item Y ser igual ao suporte de seu fecho Y'' ([Ganter e Stumme, 2003] cap. 4, Lema 15), garantindo assim, que todos os itens frequentes são determinados unicamente por conceitos formais também frequentes.

Reticulados conceituais construídos sobre itens frequentes são chamados reticulados conceituais iceberg[10]. Note que nesse caso o reticulado obtido pode ser um semi-reticulado, ou seja, um reticulado parcial no qual apenas os conceitos mais relevantes/importantes são representados. Isso ocorre, pois o suporte da intenção dos conceitos é uma função decrescente, i.e., para dois conceitos formais (X_1,Y_1) e (X_2,Y_2) em que $(X_1,Y_1) \leqslant (X_2,Y_2)$, $sup(Y_1,G) \leqslant sup(Y_2,G)$.

Para construir reticulados conceituais iceberg pode-se utilizar qualquer algoritmo que construa o reticulado conceitual em um processo *top-down*, bastando calcular a frequência da intenção de cada novo conceito criado e aplicar a definição de conceito frequente (Definição 10). Carpineto e Romano [2004a] apresentam um adaptação do algoritmo *Next Neighbours* (Subseção 3.5.1) chamada *Frequent Next Neighbours* que realiza esse processo. Um algoritmo mais interessante, apresentado por Stumme et al.

[10]NT: Em Inglês é utilizado o termo *iceberg concept lattices*.

[2002], é o algoritmo *Titanic*, que é um algoritmo para um problema mais geral: gerar conjuntos fechados arbitrários. Stumme et al. [2002] demonstram a utilização do algoritmo para geração de reticulados conceituais iceberg e o potencial desses reticulados em diversas aplicações tais como: análise de grandes bases de dados, extração de regras de implicação, extração de regras de associação, visualização de regras, extração de cluster conceituais, dentre outras.

Para exemplificar o método baseado em itens frequente é apresentado na Figura 3.1 o reticulado conceitual frequente do contexto formal para o exemplo dos animais (Tabela 1.1) com um suporte mínimo de 33%. Observe que, para o exemplo, foi possível selecionar os conceitos formais mais frequentes, graças ao limiar de corte passado como parâmetro pelo usuário.

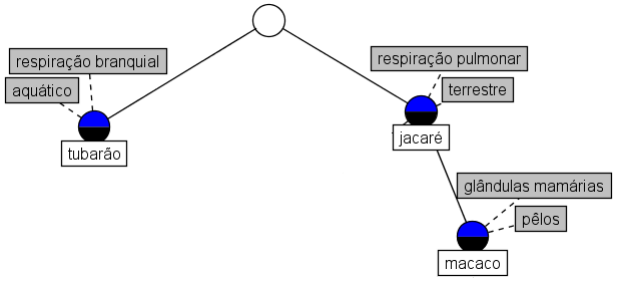

Figura 3.1: Reticulado conceitual frequente para um suporte mínimo de 33%.

Ao associar itens frequentes, e reticulados conceituais, informações importantes podem ser perdidas. Por exemplo, no contexto formal descrevendo a taxonomia dos animais o atributo “raciocínio” é necessário para distinguir os objetos “humano” e “macaco”; contudo, ele é removido pela técnica por possuir apenas uma incidência. Observe que nesse caso esse atributo é raro e possui uma alta informação, o que pode dificultar a utilização do reticulado para representação do conhecimento.

Como citado, gerar todos os conceitos formais no pior caso apresenta um custo exponencial, apesar desse comportamento ser raramente encontrado na prática, ainda assim o custo computacional pode ser proibitivo em algumas aplicações. Por outro lado, gerar todos os itens também possui um custo exponencial. Dado um conjunto de itens T existirão $2^{|T|} - 1$ subconjuntos; note que o conjunto vazio não é considerado[11]. Um

[11] O leitor mais atento irá observar que ambas as complexidades são a cardinalidade do conjunto potência, o que reforça a relação entre itens frequentes e conceitos formais frequentes.

suporte mínimo, desta forma, tem com objetivo reduzir essa complexidade selecionando apenas um subconjunto desses itens. Ao gerar apenas conceitos formais cuja intenção seja frequente, estar-se-á reduzindo a complexidade do reticulado conceitual, tornando assim, o método viável para aplicações antes consideradas impossíveis do ponto de vista computacional. É necessário ressaltar que agora existe um compromisso forte entre o suporte mínimo utilizado e a qualidade da informação ainda representada no reticulado conceitual. Além disso, segundo Stumme et al. [2002], o reticulado conceitual iceberg terá uma melhor representatividade dos dados, i.e., representará melhor a informação, quanto mais correlatos forem os dados.

Outra técnica de redução da complexidade muito interessante é baseada no conceito de estabilidade, segundo Roth et al. [2006], proposta por Kuznetsov [1990]. Ao contrário de outros métodos de redução da complexidade, a estabilidade procura criar um índice para os conceitos que indica o quanto a intenção do conceito depende do conjunto de objetos. Utilizando esse índice, pode-se assumir um limiar a partir do qual todos os conceitos formais com valor inferior são removidos, o que reforça o comentário descrito sobre a remoção do atributo "raciocínio". Para um conceito formal (X,Y) seu índice de estabilidade α será dado pela equação 3.1.

$$\alpha(X,Y) = \frac{|\{C \subseteq Y | C' = X\}|}{2^{|Y|}} \tag{3.1}$$

Contudo, dado um contexto formal e um conceito, o cálculo desse índice é #P-completo, conforme apresentado pelos autores. Note que esse custo deve-se a necessidade de gerar todos os subconjuntos $C \subseteq Y$ de cada conceito formal (X,Y). Para contornar esse problema Kuznetsov et al. [2007] apresenta duas heurísticas que proporcionam a aplicação do método. Roth et al. [2006] também apresentam um algoritmo baseado no conceito de cobertura em grafos para calcular o índice de estabilidade dos conceitos formais.

Em Roth et al. [2006] a análise formal foi utilizada em um caso prático para representar de forma significativa taxonomias. Entretanto, mesmo contextos formais relativamente pequenos apresentaram uma alta complexidade (provocado por ruído existente nos dados), o que levou os autores a aplicar o conceito de estabilidade. Os autores também combinam diagramas de linha aninhados com a poda baseada em estabilidade, criando assim representações mais adequadas para o caso prático analisado. Mais re-

centemente, Kuznetsov et al. [2007] aplicam a técnica em outras bases de dados e desenvolvem uma estabilidade dual, baseada na extensão do conceito, que agora indica o quanto a extensão depende de um atributo particular. Recentemente, Jay et al. [2008] aplicam o conceito de estabilidade e reticulados de iceberg na análise de redes sociais.

Como observado na literatura, são encontradas diversas abordagens para redução da complexidade do reticulado conceitual, o que demonstra a relevância do tema. Note que na literatura também são observadas abordagens que combinam análise formal de conceitos com outras técnicas tais como *rough theory set* [Wolskil, 2005] e teoria *fuzzy* [Zhou et al., 2007] e em seguida procuram reduzir o reticulado conceitual. Contudo, essas abordagens que estendem o arcabouço matemático da análise formal de conceitos fogem ao escopo desse trabalho.

Capítulo 4

Um Arcabouço para Desenvolvimento e Avaliação de Algoritmos da AFC

Neste capítulo é discutido um arcabouço para desenvolvimento de algoritmos da análise formal de conceitos e definido um *benchmark* para avaliação dos algoritmos.

4.1 O Arcabouço EF-Concept Analysis

A comunidade científica disponibiliza vários softwares/ferramentas para a análise formal de conceitos[Priss, 2009]. Dentre eles destacam-se os softwares ToscannaJ e Conexp. Contudo, são observados poucos softwares que proporcionam ao aluno um ambiente de desenvolvimento onde o mesmo possa implementar os algoritmos e analisar os resultados.

Para o estudo da análise formal de conceitos é preciso uma compreensão dos conceitos matemáticos nos quais ela se fundamenta e para um efetivo processo de aprendizagem, é necessário também um efetivo processo de percepção que permita uma correta reflexão por parte do aluno envolvido na atividade de aprendizagem, porém essa tarefa não é trivial. Segundo Brunner [1990], é necessário que, no processo de aprendizagem, haja uma preocupação com o contexto e as experiências do aprendiz para que o mesmo tenha vontade e seja capaz de aprender. No presente caso, o desenvolvimento e implementação dos algoritmos da análise formal podem ser apropriados para envolver o aluno no processo de aprendizagem, consolidando assim sua capacidade para o entendimento

e manipulação de algoritmos e do arcabouço matemático da análise formal.

Nesta seção, é proposto um arcabouço didático para o desenvolvimento de algoritmos da análise formal chamado *EF-Concept Analysis* (*Educational Framework for Concept Analysis*). Esse arcabouço foi desenvolvido em Java 1.6 utilizando o paradigma de orientação por objeto. Além disso, todos os serviços foram orientados por interface, de forma que o arcabouço pudesse ser estendido facilmente. Na Figura 4.1 o diagrama de pacotes do arcabouço desenvolvido pode ser observado. Ele é composto de oito pacotes com as seguintes funcionalidades:

- *io*: responsável por entrada e saída dos dados. Possui uma interface para leitura do contexto formal no formato *burmeister*, que é amplamente utilizado pela comunidade. Para exibir o reticulado conceitual é utilizado um formato XML chamado GSH (*Galicia Sub-Hierarchy*) lido pelo software Galacia[1]. Observe que outros formatos podem sem ser facilmente utilizados, simplesmente implementando-se as interfaces definidas.

- *set*: possui uma interface definindo operações sobre conjunto. Também possui classes implementando conjuntos simples e ordenados.

- *statistic*: possui um conjunto de classes e métodos para colher estatísticas do reticulado conceitual e para análise da relação entre os dados utilizados para construir o reticulado conceitual e um conjunto de regras extraídas do reticulado.

- *exception*: conjunto de exceções lançadas pelo arcabouço.

- *context*: define uma interface para um contexto formal e uma estrutura de dados baseada em cadeia de bits para representar o contexto. Também são definidos tipos abstratos de dados para objetos, atributos e conceitos formais.

- *util*[2]: conjunto de classes úteis, dentre elas destacando-se uma classe para produção de contextos formais a partir de dados reais.

[1] Disponível em: http://www.iro.umontreal.ca/~galicia/.

[2] Acrônimo do Inglês para *utilities*, que pode ser compreendido como utilitários ou simplesmente úteis.

- *algorithms*: conjunto de algoritmos e exemplos de utilização do arcabouço. O arcabouço possui implementados os algoritmos *Next Closure*, *Concepts Cover*, *Next Neighbours*, *Frequent Next Neighbours*, *Godin* e *Find Implications*.

- *benchmark*: interfaces e classes para produção de contextos formais sintéticos.

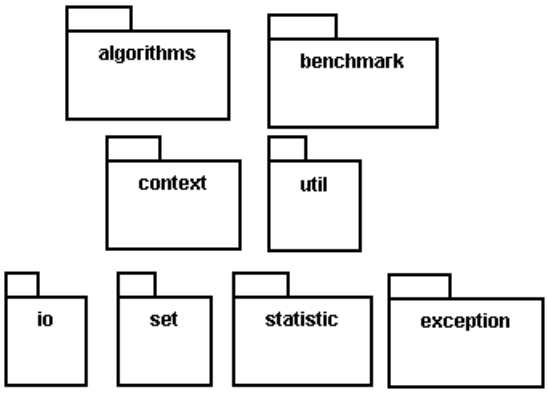

Figura 4.1: Diagrama de pacotes do arcabouço *EF-Concept Analysis*.

Na Figura 4.2 pode ser observado o diagrama de classes simplificado dos principais algoritmos desenvolvidos e das estruturas de dados e interfaces envolvidas. Pode-se observar que as estruturas se interligam mostrando a relação entre as diversas entidades. Por exemplo, a relação entre reticulado (*lattice*), reticulado conceitual (*concept lattice*) e semi-reticulado (*semi-lattice*), mostrando que as duas últimas classes estendem a primeira.

Para exemplificar a utilização do arcabouço *EF-Concept Analysis* é apresentado no código 4.1 uma codificação parcial em Java do algoritmo *Next Neighbours*. Na Subseção 3.5.1 (Algoritmo 7) a função principal do algoritmo pode ser observada; note que a codificação em Java apresentada, utilizando o arcabouço, é bem similar ao algoritmo em alto nível. Essa relação próxima entre uma representação do algoritmo em alto nível e uma codificação em uma linguagem de programação é muito útil em um processo de aprendizagem, pois facilita o desenvolvimento e estudo do algoritmo.

O objetivo dessa seção foi apresentar o arcabouço *EF-Concept Analysis*; contudo cabe ressaltar que esse não é o foco principal desse trabalho. Porém, o arcabouço foi apresentado com o intuito de formalizar o trabalho desenvolvido e proporcionar sua

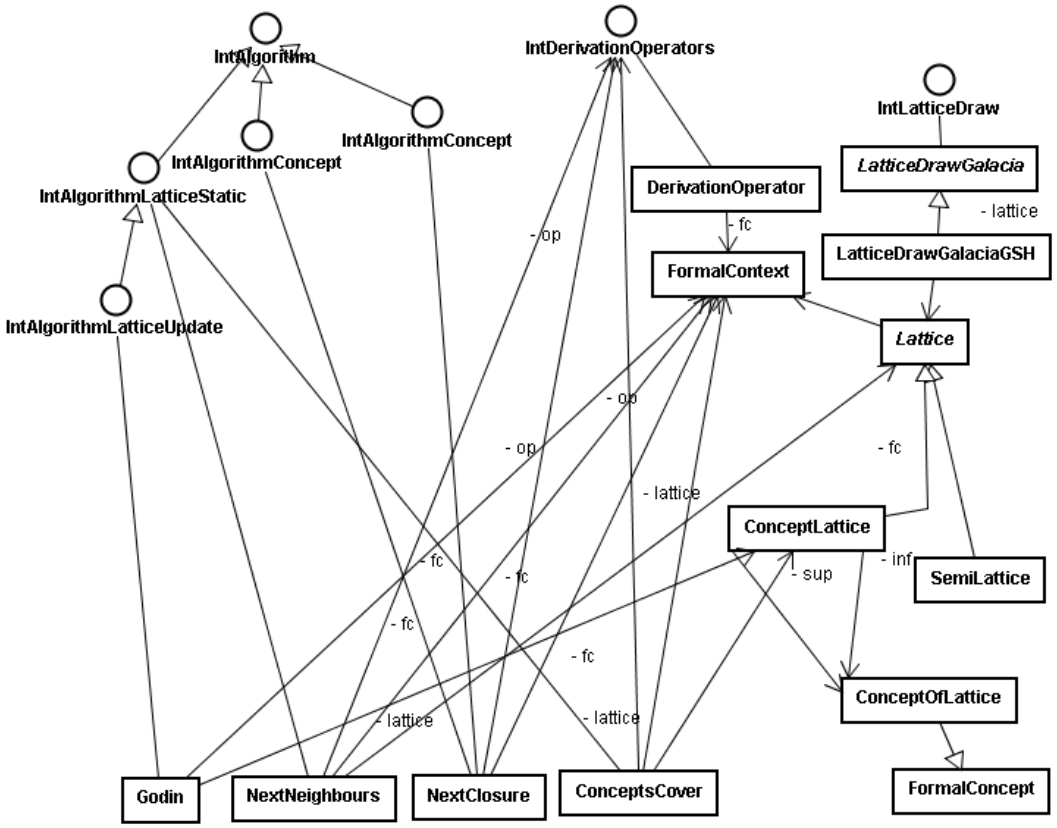

Figura 4.2: Diagrama de classe com as principais estruturas e algoritmos.

expansão futura, já que são observados poucos arcabouços dessa natureza para a análise formal de conceitos. Assim, esse arcabouço, apesar de ainda simples, pode ser útil para aprendizado e uso dos algoritmos, o que justifica torná-lo disponível.

4.2 Benchmark para Avaliação dos Algoritmos e Paradigmas

Um problema constantemente comentado na literatura é o do teste de desempenho dos algoritmos para construção do reticulado conceitual. Como observado na Seção 3.2 existe uma grande quantidade de algoritmos e paradigmas disponíveis. Assim, alguns problemas eminentes são: qual algoritmo é mais adequado para gerar os conceitos formais e o reticulado conceitual? Como comprovar sua eficiência? Qual abordagem é mais adequada para construção do reticulado conceitual (gerar os conceitos formais e

```
protected void nextNeighbours() throws SetException{
//C - conceitos gerados
Hashtable<Long,ConceptOfLattice> c= new Hashtable<Long,ConceptOfLattice>();
IntSet<ConceptOfLattice> currentLevel = new ArraySet<ConceptOfLattice>();
IntSet<ConceptOfLattice> nextLevel ;
IntSet<ConceptOfLattice> lowerNeigbours;

//gera o conceito supremo (G,G')
ConceptOfLattice supremo = new ConceptOfLattice(
        ConceptOfLattice.SUPREMO,
        this.contextoFormal.getObjetos(),
        this.operadorDerivacao.derivarObjeto(this.contextoFormal.getObjetos()));

//c = c {(G,G')}
c.put(supremo.getId(),supremo);
//currentLevel = currentLevel U {(G,G')}
currentLevel.inserir(supremo);
//adiciona o supremo na estrutura do reticulado
((ConceptLattice)reticulado).setSupremo(supremo);

//enquanto currentLevel nao vazio
while(!currentLevel.vazio()){
        //nextLevel=0
        nextLevel = new ArraySet<ConceptOfLattice>();
        //cada (X,Y) PER currentLevel
        for(ConceptOfLattice c1 : currentLevel.getColecao()){
                //lowerNeigbours = findLowerNeibours((X,Y));
                lowerNeigbours = findLowerNeibours(c1);
                //cada (X1,Y1) PER lowerNeigbours
                for(ConceptOfLattice c2 : lowerNeigbours.getColecao()){
                        //(X1,Y1) NOT PER C
                        if(!c.containsKey(c2.getId())){
                                //C = C U (X1,Y1)
                                c.put(c2.getId(),c2);
                                //nextLevel = nextLevel U (X1,Y1)
                                nextLevel.inserir(c2);
                                reticulado.adicionarConceito( c2);
                        }//
                        //add edge (X1,Y1) -> (X,Y) to E
                        c2.addSuperior(c1); //adicionar current no inferior de lower
                        c1.addInferior(c2); //adicionar lower no superior de current
                }//end lower
        }//end current
        currentLevel = nextLevel;
}//end while
}//end nextNeighbours()
```

Código 4.1: Função Next Neighbours codificada em Java.

ordená-los, gerar os conceitos e o reticulado conceitual em um único processo, utilizar um paradigma incremental, etc)?

Kuznetsov e Obiedkov [2002] comentam que existe uma dificuldade muito grande em comparar os algoritmos pela inexistência de um padrão de testes. Recentemente foi

proposto um *workshop*[3] com o objetivo de comparar o desempenho dos algoritmos da análise formal de conceitos para três problemas clássicos:

- gerar o conjunto de todos os conceitos formais;
- gerar o reticulado conceitual;
- extrair uma base de implicações (conjunto de regras) do reticulado conceitual mínima, completa e não redundante conhecida na literatura por *duquenne-guigues base* ou *steam base*.

Esse *workshop* demonstra como atualmente é considerado importante o processo de comparação dos algoritmos da análise formal, o que justifica a proposta aqui apresentada.

De fato, observa-se uma grande discrepância no processo de comparação dos algoritmos na literatura [Guénoche, 1990; Godin et al., 1995, 1991; Kuznetsov e Obiedkov, 2001; Fu e Nguifo, 2004; Arévalo et al., 2007]. Kuznetsov e Obiedkov [2002] também comentam que existe a necessidade da comunidade definir um conjunto de contextos formais bem definidos, além da definição de um padrão de testes. Assim, nessa seção é apresentado um conjunto de contextos formais sintéticos e reais, que procuram estabelecer um padrão de comparação dos algoritmos da análise formal, criando assim um *benchmark* de avaliação dos algoritmos.

4.2.1 Benchmark Sintético

O número de conceitos formais de um contexto formal é influenciado pelo número de objetos, atributos, densidade e pela forma de distribuição da densidade. Portanto, um conjunto de contextos formais sintéticos para avaliação dos algoritmos e paradigmas de construção do reticulado conceitual deve considerar esses parâmetros. Na literatura pouco é observado sobre contextos formais sintéticos. Em geral, os trabalhos que propõem/utilizam esse tipo de contexto procuram variar esses parâmetros, como os trabalhos apresentados por Arévalo et al. [2007], Kuznetsov e Obiedkov [2001] e Kuznetsov e Obiedkov [2002], dentre outros.

[3] http://www.iccs09.org/forum/

Uma segunda opção muito utilizada na literatura é o software *IBM synthetic data generator*[4], o qual implementa a metodologia proposta por Agrawal et al. [1996]. Em síntese, Agrawal et. al. propõem uma base de dados sintética voltada para mineração de dados, mais precisamente para extração de regras de associação, que espelha-se no comportamento de bases de dados transacionais. Uma crítica quanto à utilização dessa base de dados sintética é o fato de seu comportamento ser característico de um tipo de aplicação. Ao mudar o foco da aplicação, por exemplo, para recuperação de informação, bioinformática, redes sociais, etc, o comportamento, a distribuição de incidências no contexto formal, será diferente. Desta forma, esse trabalho tem como objetivo a proposta de um *benchmark* sintético de uso geral, que possa ser utilizado para testes de algoritmos. Por isso, são propostos vários tipos de contextos formais sintéticos que exploram diferentes características.

Antes de apresentar as características dos contextos formais propostos é necessário realizar algumas considerações. Primeiramente, os valores sugeridos (número de objetos, atributos e densidade) foram baseados no desempenho observado para o arcabouço *EF-Concept Analysis* e podem ser considerados valores típicos para um contexto formal. Segundo, as palavras "muito" e "pouco" ou até mesmo a definição de um valor fixo contém um alto grau de subjetividade. Um algoritmo projetado para ter alto desempenho pode superar alguns dos contextos propostos. Entretanto, ainda assim é possível ter alguma medida (tempo, memória ocupada, etc) para um contexto formal definido/conhecido, que permita realizar comparações; além disso, pode-se perfeitamente alterar os valores sugeridos.

Em resumo, os valores sugeridos para os contextos formais não são relevantes, e sim o processo de comparação proposto, o qual procura cobrir uma gama a mais ampla possível de situações encontradas/observadas nos algoritmos. Em geral, os testes de desempenho dos algoritmos da análise formal observados na literatura não apresentam essa preocupação. Por exemplo, não é comum realizar testes utilizando contextos formais com uma quantidade maior de objetos que atributos. Contudo, como citado por Fu e Nguifo [2003] frequentemente é observado um desempenho diferente para contextos formais com diferentes características. O *benchmark* sintético proposto procura superar esses pontos falhos observados na literatura. Segue a proposta de aspectos para os testes e valores de parâmetros respectivamente:

[4]Uma versão do software pode ser encontrada em http://www.cs.rpi.edu/~zaki/software/.

1. Explore o pior caso dos algoritmos (contextos formais com o mesmo número de objetos e atributos e apenas a diagonal principal não preenchida.). Um conjunto de valores típicos para esses testes é $|M| = |G| = 2, 3, 4, ..., n$.

2. Crie contextos formais com várias densidades de I, sendo seus elementos distribuídos aleatoriamente, por exemplo, $|I| = 5, 10, 15, ..., 95$ (%). Para cada valor de densidade utilize poucos objetos e muitos atributos e muitos objetos e poucos atributos. Sugere-se os seguintes valores:

   - poucos objetos e muitos atributos $|G| = 20$ e $|M| = 50$;
   - muitos objetos e poucos atributos $|G| = 50$ e $|M| = 20$.

3. Crie contextos formais com várias densidades de I, sendo seus elementos distribuídos aleatoriamente, e mesmo número de atributos por objetos, por exemplo, $|I| = 5, 10, 15, ..., 95(\%)$. Para cada valor de densidade utilize poucos objetos e muitos atributos e muitos objetos e poucos atributos. Sugere-se os seguintes valores:

   - poucos objetos e muitos atributos $|G| = 20$ e $|M| = 50$;
   - muitos objetos e poucos atributos $|G| = 50$ e $|M| = 20$.

4. Varie o número de objetos para uma quantidade fixa de atributos e uma densidade de I fixa, cujos elementos sejam distribuídos aleatoriamente. Por exemplo, utilize $|G| = 20, 40, 60, ..., n$, $|M| = 20$ e $|I| = 50\%$.

5. Varie o número de atributos para uma quantidade fixa de objetos e uma densidade de I fixa, cujos elementos sejam distribuídos aleatoriamente . Por exemplo, utilize $|M| = 20, 40, 60, ..., n$, $|G| = 20$ e $|I| = 50\%$.

O primeiro item, pior caso (Seção 3.1), procura levar o algoritmo/paradigma ao extremo. Alguns autores como Kuznetsov e Obiedkov [2002]; Godin et al. [1995] e Carpineto e Romano [2004a] comentam que esse comportamento é raramente observado em problemas práticos. Contudo, saber qual é o limite superior de um algoritmo é extremamente relevante por revelar, por exemplo, um limiar a partir do qual o uso apenas de memória primária não é mais possível.

O parâmetro que mais influência no número de conceitos formais é a densidade (número de incidências $|I|$) e como ela está distribuída; desta forma, propõe-se a variação da densidade. Em relação à distribuição da densidade de I, sugere-se a utilização de uma distribuição aleatória na qual cada objeto pode possuir qualquer número de atributos, e outra, na qual cada objeto possui o mesmo número de atributos. Como a quantidade de objetos e atributos também irá influenciar no número de conceitos formais, propõe-se a combinação desses parâmetros da seguinte forma: muitos objetos e poucos atributos, muitos atributos e poucos objetos.

Por último propõe-se a variação do número de objetos e atributos para uma densidade. Note que a variação do número de objetos proposta é maior, já a variação do número de atributos é menor e com uma granularidade também menor. Essa diferença espelha-se em contextos reais, que frequentemente apresentam uma quantidade maior de objetos que atributos.

Cabe ressaltar que os testes propostos não levam em consideração a possibilidade dos contextos formais apresentarem objetos com os mesmos atributos ou atributos com os mesmos objetos (linhas ou colunas idênticas). Em contextos reais essa característica pode ser encontrada; quando isso ocorrer, conforme citado na Seção 3.8, pode se limpar o contexto formal.

Na próxima subseção são apresentados contextos formais reais, que juntamente com os contextos sintéticos formam o *benchmark* definido para avaliação dos algoritmos e paradigmas de construção do reticulado conceitual.

4.2.2 Benchmark Real

Uma questão desafiadora é: como escolher um conjunto de dados reais para avaliação dos algoritmos que possam ser aceitos pela comunidade científica? Nos trabalhos observados na literatura, cujo objetivo é apresentar algum algoritmo e/ou realizar comparações, é constantemente observado críticas quanto às características das bases de dados reais utilizadas. Em geral observam-se críticas sobre a qualidade dos dados (dimensões, número de registros), bases não disponíveis, falta de descrição dos contextos formais construídos, dentre outras.

Assim, como parte do *benchmark* proposto para avaliação dos algoritmos e paradigmas de construção do reticulado conceitual é apresentado e caracterizado, nessa Subseção, um conjunto de contextos formais construídos a partir de dados reais.

Foram selecionadas quatro bases de dados clássicas da UCI (*Machine Learning Repository*)[Asuncion e Newman, 2009], são elas: *Iris Data Set, Wine Data Set, Yeast Data Set* e *Water Treatment Plant Data Set*[5]. As três primeiras bases estão entre as dez principais bases de um total de 177 bases de dados segundo índice de popularidade mantido pelo repositório desde 2007. As bases de dados foram selecionadas seguindo o critério de popularidade, número de registros e número de atributos, sendo o critério de utilização pela comunidade considerado o fator mais relevante.

Nas próximas subseções as bases de dados são apresentadas e o processo de construção de cada contexto formal é detalhado.

4.2.2.1 Base de Dados Íris

A base de dados Íris é considerada uma base dados simples, que é composta de 150 registros, 4 parâmetros descritivos contínuos e um parâmetro classificador composto de 3 classes. O objetivo do conjunto de dados é classificar a planta Íris, que é um gênero de planta com flor. Segundo a UCI[Asuncion e Newman, 2009], e as descrições constantes na base de dados, ela é uma das bases de dados mais conhecidas na literatura. A base Íris não apresenta dados ausentes e é perfeitamente balanceada, já que cada classe possui exatamente 33.33% registros.

Para construção do contexto formal utilizou-se o seguinte procedimento: cada classe do parâmetro classificador gerou um atributo no contexto formal; já os parâmetros descritivos foram divididos em 2 ou 3 intervalos. O número de intervalos e os pontos de corte[6] foram determinados através da análise dos gráficos de distribuição de cada parâmetro (Anexo A). Analisando a distribuição dos parâmetros *petal length (cm)* e (Figura A.3) *petal width (cm)* (Figura A.4) percebe-se a formação de 3 grupos, neste caso optou-se pela criação de três atributos para o contexto formal (2 pontos de corte). Desta forma, essa característica continua preservada no contexto formal. Já os demais parâmetros foram divididos em 2 intervalos com base em análise semelhante.

Os pontos de corte foram escolhidos de acordo com a relação de incidência criada no atributo. Procurou-se gerar intervalos com aproximadamente a mesma quantidade

[5] As bases de dados serão referenciadas no texto por Íris, Wine, Yeast, e Water, respectivamente.

[6] Na literatura especializada em processos de discretização é utilizado o termo ponto de corte (do Inglês *cut point*) para determinar o valor numérico que divide um conjunto de valores reais em dois conjuntos.

de incidências. Esse procedimento evita a geração de atributos $m \in M$ nos quais $|m'|$ é muito discrepante, o que resultaria em um contexto formal muito desbalanceado.

Na Tabela 4.1 os pontos de corte aplicados na construção de cada atributo da análise formal, a partir dos parâmetros da base de dados Íris, podem ser observados[7]. Note que não são encontradas grandes discrepâncias entre os números de incidências dos atributos criados.

Tabela 4.1: Atributo criado para o contexto formal, intervalos e incidência para base de dados Íris.

Atributo		
Parâmetro	**Intervalo**	**Nº de incidências**
sepal length	[4.3 - 6.1>	89
sepal length	[6.1 - 7.9]	61
sepal width	[2 - 3.1>	83
sepal width	[3.1 - 4.4]	67
petal length	[1 - 2>	50
petal length	[2 - 5>	54
petal length	[5 - 6.9]	46
petal width	[0.1 - 0.6>	49
petal width	[0.6 - 1.8>	55
petal width	[1.8 - 2.5]	46
Classe 1	Íris setosa	50
Classe 2	Íris versicolor	50
Classe 3	Íris virginica	50
Média		57.6
Desvio padrão		13.9

Dentre as bases de dados reais propostas, como poderá ser observado, a base Íris possui o menor número de conceitos formais, apenas 104 conceitos. Essa quantidade de conceitos formais permite uma análise visual do reticulado conceitual, por exemplo, através de um diagrama de linhas. Muitas aplicações fazem uso visual do reticulado; desta forma, essa base de dados torna-se muito útil. Na Figura 4.3 o reticulado conceitual para base Íris pode ser observado.

[7]Utilizou-se a notação "[]" e "[>" para designar um intervalo fechado e um intervalo fechado a esquerda e aberto a direita, respectivamente.

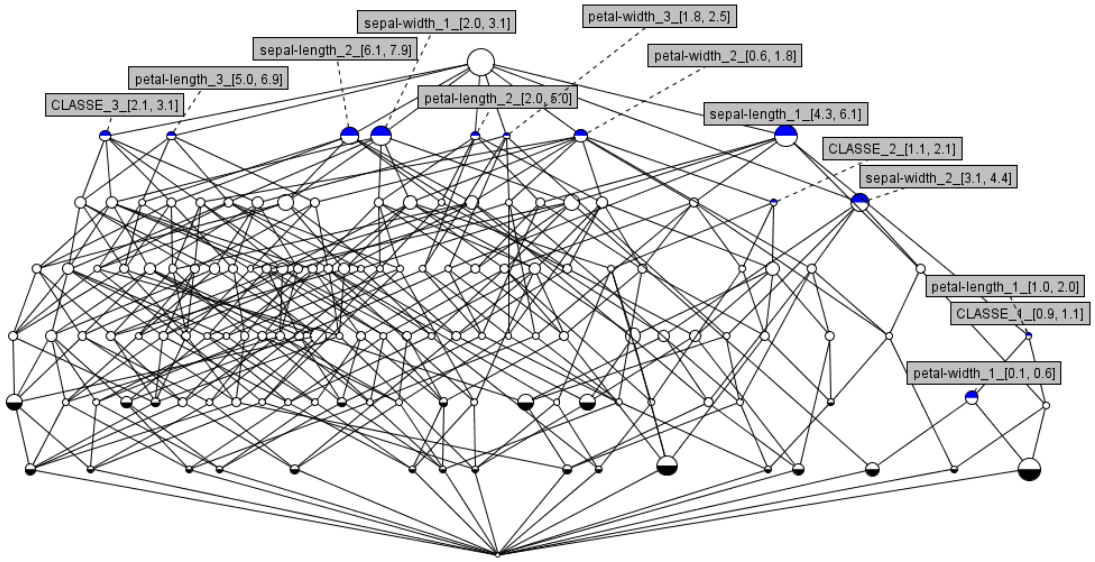

Figura 4.3: Diagrama de linhas para base de dados Íris.

4.2.2.2 Base de Dados Wine

A base de dados Wine tem como objetivo classificar três diferentes vinhos a partir de um conjunto de parâmetros obtidos através de análises químicas[Asuncion e Newman, 2009]. A base não apresenta dados ausentes; ela é composta de 13 parâmetros descritivos (números reais) e um classificador, com um total de 178 registros.

Para a base de dados Wine todos os parâmetros descritivos foram divididos em dois intervalos, gerando dois atributos cada um para o contexto formal. Novamente, os pontos de corte foram selecionados através da análise visual dos gráficos de distribuição de cada parâmetro (Anexo A), levando em consideração um número equilibrado de incidências em cada partição. Já o atributo classificador foi dividido em 3 intervalos, um para cada classe. Na Tabela 4.2 os atributos gerados, os intervalos de cada atributo e a incidência de cada atributo podem ser observados.

O contexto formal produzido para a base de dados Wine possui 29032 conceitos formais. Note que essa quantidade de conceitos formais não permite mais uma análise visual do reticulado conceitual, porém ela é adequada para uma comparação dos algoritmos e paradigmas de construção do reticulado conceitual.

Tabela 4.2: Atributo criado para o contexto formal, intervalos e incidência para base de dados Wine.

Atributo		
Parâmetro	**Intervalo**	**Nº de Incidência**
Alcohol	[11.03-12.93>	82
Alcohol	[12.93 - 14.83]	96
Malic acid	[0.74 - 2>	100
Malic acid	[2 - 5.8]	78
Ash	[1.36 - 2.35>	83
Ash	[2.35 - 3.23]	95
Alcalinity of ash	[10.6 - 19>	77
Alcalinity of ash	[19 - 30]	101
Magnesium	[70 - 100>	97
Magnesium	[100 - 162]	81
Total phenols	[0.98 - 2.44>	96
Total phenols	[2.44 - 3.88]	82
Flavanoids	[0.33 - 2.2>	93
Flavanoids	[2.2 - 5.08]	85
Nonflavanoid phenols	[0.13 -0.36>	96
Nonflavanoid phenols	[0.36 -0.66]	82
Proanthocyanins	[0.41 - 1.6>	94
Proanthocyanins	[1.6 - 3.58]	84
Color intensity	[1.28 - 5.0>	95
Color-intensity	[5.0 - 13]	83
Hue	[0.48 - 1.0>	94
Hue	[1.0 - 1.71]	84
OD280/OD315 of diluted wines	[1.27 - 2.76>	84
OD280/OD315-of-diluted-wines	[2.76 - 4]	94
Proline	[278 - 678>	91
Proline	[678 - 2680]	87
Classe 1	1	59
Classe 2	2	71
Classe 3	3	48
Média		85.9
Desvio padrão		11.8

4.2.2.3 Base de Dados Yeast

A terceira base de dados escolhida é a base de dados Yeast[Asuncion e Newman, 2009]. Ela tem como objetivo a classificação da localização de proteínas na célula. A base Yeast não apresenta dados ausentes e possui 9 parâmetros, 8 descritivos e 1 classificador, e 1484 registros. O parâmetro classificador possui 10 classes não balanceadas

(Figura A.28), isto é, existe uma grande discrepância entre a quantidade de registros em cada classe. Dos 8 parâmetros descritivos, 7 possuem valores reais e 1 valor binário (parâmetro *ERL*).

Novamente foi aplicado um processo de discretização similar ao utilizado nas demais bases; dividindo os parâmetros descritores em 2 ou 3 intervalos e o parâmetro classificador em 1 intervalo para cada classe. Apenas o parâmetro *POX* foi divido em 3 intervalos por apresentar 3 grupos na análise do gráfico de dispersão (Figura A.25). Na Tabela 4.3 os atributos criados para o contexto formal, os respectivos intervalos e o número de incidências podem ser observados. Já em relação ao número de conceitos formais, após a criação do contexto formal, foi obtido um total de 1930 conceitos.

4.2.2.4 Base de Dados Water

A última base de dados considerada é a base de dados Water[Asuncion e Newman, 2009], que possui registros de falhas em uma instalação de tratamento de água residual urbana. A base é composta de 380 registros e 38 parâmetros, que possuem apenas valores reais. A base apresenta dados ausentes, aproximadamente 27.7%, os quais foram removidos.

Como todos os parâmetros da base de dados são valores contínuos, optou-se por um processo de discretização equidistante e com um número fixo de 1 ponto de corte para cada parâmetro, gerando-se 76 atributos no contexto formal. Esse contexto formal construído apresentou um total de 8940648 conceitos formais, e como poderá observado no Capítulo 6, a geração do reticulado conceitual nesse caso nem sempre será possível para o arcabouço *EF-Concept Analysis*.

4.2.2.5 Resumo das Características das Bases de Dados Reais

Na Tabela 4.4 um resumo das características das bases de dados reais que compõem o *benchmark* proposto pode ser observado. Note que as bases possuem diferentes características.

Durante o processo de discretização para construção do contexto formal pode-se produzir objetos que possuem os mesmos atributos e atributos que possuem os mesmos objetos, isto é, linhas e colunas idênticas. Nesse caso, conforme discutido na Subseção 3.8.1, pode-se limpar o contexto formal. Na Tabela 4.5 um resumo das características das bases de dados após a remoção dos objetos e atributos idênticos pode ser observado.

Tabela 4.3: Atributo criado para o contexto formal, intervalos e incidência para base de dados Yeast.

Atributo		
Parâmetro	**Intervalo**	**Nº de Incidência**
MCG	[0.11, 0.555>	1042
MCG	[0.555, 1.0]	442
GVH	[0.13, 0.565>	1088
GVH	[0.565, 1.0]	396
ALM	[0.21, 0.605>	1373
ALM	[0.605, 1.0]	111
MIT	[0.0, 0.5>	1375
MIT	[0.5, 1.0]	109
ERL	[0.5, 0.75>	1470
ERL	[0.75, 1.0]	14
POX	[0.0, 0.276>	1469
POX	[0.2766, 0.553>	4
POX	[0.553, 0.83]	11
VAC	[0.24, 0.435>	39
VAC	[0.435, 0.63]	1445
NUC	[0.0, 0.5>	1411
NUC	[0.5, 1.0]	73
Classe 1	mitochondrial	244
Classe 2	nuclear	429
Classe 3	cytosolic or cytoskeletal	463
Classe 4	membrane protein, cleaved signal	44
Classe 5	extracellular	35
Classe 6	membrane protein, uncleaved signal	51
Classe 7	membrane protein, no N-terminal signal	163
Classe 8	vacuolar	30
Classe 9	peroxisomal	20
Classe 10	endoplasmic reticulum lumen	5
Média		494.6
Desvio padrão		578.8

Note que, conforme discutido na Subseção 3.8.1, o número de conceitos formais não é alterado, como pode ser observado na última linha da tabela.

Tabela 4.4: Resumo das características das bases de dados reais que compõem o *benchmark*.

	Contextos Formais			
	Íris	**Wine**	**Yeast**	**Water**
Número de objetos	150	178	1484	380
Número de atributos	13	29	27	76
Número médio de objetos por atributos	5	14	9	38
Densidade	38.4 %	48%	33%	50%
Número de Conceitos Formais	104	29032	1930	8940648

Tabela 4.5: Resumo das características das bases de dados reais que compõem o *benchmark* após redução.

	Contextos Formais			
	Íris	**Wine**	**Yeast**	**Water**
Número de objetos	18	132	110	316
Número de atributos	12	29	27	76
Número médio de objetos por atributos	4.8%	14%	9%	34.9%
Densidade	40.3%	48.2%	33.3%	52.1%
Número de Conceitos Formais	104	29032	1930	8940648

Capítulo 5

Redução da Complexidade do Reticulado Conceitual

Neste capítulo é discutida uma técnica para avaliação dos métodos de redução da complexidade do reticulado conceitual e apresentadas duas propostas para redução da complexidade do reticulado.

5.1 Avaliação das Técnicas de Redução da Complexidade

Um desafio observado é com relação à avaliação das técnicas de redução da complexidade do reticulado conceitual. Como mensurar a qualidade da informação representada no reticulado? Após a aplicação de alguma técnica de redução da complexidade do reticulado conceitual, como saber qual a relevância da informação perdida?

As técnicas de redução da complexidade do reticulado conceitual analisadas (Seção 3.8) são em geral utilizadas para uma aplicação específica. Além disso, são raras as comparações entre as técnicas, fato que torna difícil a escolha de uma delas.

Cabe ressaltar que a proposta apresentada nessa seção não considera as técnicas de redução da complexidade do reticulado conceitual que apenas eliminam informação redundante (Seção 3.8). Nessa subseção é apresentado um método de avaliação da qualidade do conhecimento representado no reticulado conceitual. O método proposto é independente de qualquer possível aplicação do reticulado e procura avaliar a perda de informação ocasionada pelo processo de redução da complexidade. Também serão apresentadas métricas que permitem mensurar essa perda.

Inicialmente, considere uma base de dados T, um contexto formal $K = (G,M,I)$ construído a partir da base de dados T e um reticulado conceitual $L = B(G,M,I)$ ori-

ginado do contexto formal K. Considere também que o processo de construção do contexto é capaz de representar adequadamente a informação contida na base de dados T[1]. Suponha também que não foi utilizada nenhuma técnica de redução da complexidade no contexto formal ou no reticulado conceitual. Nesse caso, todo conhecimento do contexto formal K foi representado no reticulado conceitual L.

Por outro lado, caso alguma técnica de redução seja aplicada no contexto formal K ou no reticulado conceitual L, parte do conhecimento inicialmente representado no contexto pode ser perdido. Note que as técnicas de redução da complexidade utilizam algum limiar, a partir do qual um conjunto de conceitos formais, ou parte da informação contida nos conceitos formais, são eliminados (Seção 3.8). Dessa forma, é necessário relacionar a informação inicialmente representada no contexto formal com a informação representada no reticulado conceitual. Propõe-se para análise dessa relação o estudo de um conjunto de regras que podem ser extraídas do reticulado conceitual.

Contudo, é necessário definir qual conjunto de regras é mais adequado para esse estudo. A partir de um reticulado conceitual podem ser extraídas regras determinísticas e probabilísticas. As regras determinísticas podem ser regras de implicação ou dependências funcionais; já as regras probabilísticas dividem-se em regras de classificação e regras de associação [Vimieiro, 2007].

Para o relacionamento entre o contexto formal K e o reticulado conceitual L as regras mais apropriadas são as determinísticas, pois nesse caso não existe uma condição probabilística para a validade das regras. Dentre as regras determinísticas, as regras de implicação são indicadas. Apesar da existência de uma forte relação entre regras de implicação e dependências funcionais, as dependências são mais indicadas no contexto de banco de dados e são regras relacionando atributos multivalorados [Vimieiro, 2007]. Outra vantagem na utilização das regras de implicação é o fato delas serem extraídas diretamente da hierarquia dos conceitos formais. Portanto, para esse estudo sugere-se a utilização de regras de implicação. Segue a definição formal de uma regra de implicação [Godin e Missaoui, 1994]:

Definição 11. *Uma regra de implicação é uma expressão $P \rightarrow Q$, em que $P, Q \subseteq M$.*

Uma regra de implicação é lida da seguinte forma: **se** P **então** Q. Como comentado, existe uma relação entre regras de implicação e o reticulado conceitual. Uma implicação

[1] O processo de construção do contexto formal não é considerado, pois diversos métodos podem ser aplicados, obtendo resultados diferentes em relação à qualidade da informação representada.

$P \rightarrow Q$ ocorre no contexto formal K e, consequentemente, no reticulado conceitual L se, e somente se, $P' \subseteq Q'$. Em outras palavras: todo objeto que possui os atributos de P possui os atributos de Q. Por exemplo: a regra de implicação $\{penas\} \rightarrow \{terrestre,$ *respiração pulmonar*$\}$ segue do exemplo sobre taxonomia dos animais (Figura 1.1). Ela diz que: todo animal que possui *penas*, é *terrestre* e possui *respiração pulmonar*. Isto segue do fato de que $\{penas\}' = \{coruja\}$ e $\{terrestre,$ *respiração pulmonar*$\}' = \{$*jacaré, sapo, humano, macaco, coruja*$\}$ (Tabela 1.1). Na Subseção 5.1.1 o processo de extração de regras de implicação será tratado em mais detalhes.

Associado a um conjunto de regras tem-se o conceito de inferência. Seja F um conjuntos de regras válidas em um contexto formal, em que $F = \{A \rightarrow B, B \rightarrow C\}$. Nesse caso, pode-se inferir por transitividade que $A \rightarrow C$, em outras palavras, $A \rightarrow C$ é dedutível a partir de F ($F \models A \rightarrow C$).

Note que a extração de todas as regras válidas em um contexto formal é impraticável. Contudo, pode-se extrair um conjunto de regras menor, a partir do qual o conjunto completo de regras possa ser deduzido. Suponha agora dois conjuntos de regras F e G válidas em um contexto formal. O conjunto F será *completo* se toda regra válida em um contexto for implicada logicamente em F; quando $F^+ = G^+$ tem-se dois conjuntos F, G *equivalentes*, em que F^+ é o conjunto de todas as regras implicadas logicamente de F ($F^+ = \{X \rightarrow Y | F \models X \rightarrow Y\}$). Por último, tem-se o conceito de conjunto *redundante*; um conjunto será dito redundante se existir $X \rightarrow Y \in F$, tal que $F^+ = (F - \{X \rightarrow Y\})^+$.

Como apresentado, as regras de implicação podem ser utilizadas para análise da relação entre o contexto formal K e o reticulado conceitual L. Entretanto, qual conjunto de regras é mais adequado? A resposta inicial para esta pergunta é um conjunto mínimo, isto é, sem regras "descartáveis" e, consequentemente, com regras não redundantes. Entretanto, gerar esse conjunto de regras apresenta um custo computacional elevado [Carpineto e Romano, 2004a]; em geral, quanto mais restrito é um conjunto de regras, mais complexo é sua obtenção. Por outro lado, gerar um conjunto de regras sem restrições, isto é, todas as regras válidas em um contexto formal também é altamente custoso. Esse custo deve-se ao fato do número de regras válidas, assim como o número de conceitos formais, apresentar um comportamento exponencial [Ganter e Stumme, 2003].

Desta forma, propõe-se a geração de um conjunto de regras reduzidas à esquerda, apesar de ainda redundante. Um conjunto de regras F é chamado *reduzido à esquerda*,

se toda regra $X \rightarrow Y \in F$ é reduzida à esquerda. A regra $X \rightarrow Y$ é dita ser *reduzida à esquerda* se, para $X = \{a\} \cup Z$, $(F - \{X \rightarrow Y\}) \cup \{Z \rightarrow Y\}$ não é equivalente a F. Como poderá ser observado na Subseção 5.1.1 essa restrição evita regras "redundantes" originadas a partir de um conceito formal.

Definidos a extração de regras de implicação e o tipo de restrição imposta ao conjunto de regras, pode-se agora relacionar o contexto formal K e o conjunto de regras, que será denotado por $\varphi = \{f_1, f_2, ..., f_m\}$. Entre o contexto formal e o conjunto de regras existe um *grau de equivalência*; em outras palavras, a informação existente no conjunto de regras extraído do reticulado conceitual é equivalente à informação existente no contexto formal original, mas apenas em certo grau.

A Figura 5.1 apresenta um diagrama da metodologia. Inicialmente, tem-se uma representação dos dados pelo contexto formal original; novamente, note que a perda de informação ocasionada por um eventual processo de discretização não é considerada. Em seguida tem-se um contexto formal reduzido, a partir do qual é gerado um reticulado conceitual e, finalmente, desse reticulado um conjunto de regras de implicação é obtido. Já entre o contexto formal original e o conjunto de regras existirá um grau de equivalência, que expressa o quanto o conjunto de regras representa o contexto formal original. Note que, quando não for utilizada nenhuma técnica de redução da complexidade para obtenção do contexto formal reduzido ou do reticulado conceitual, esse grau de equivalência será máximo. Por outro lado, quando algum método for aplicado, o grau de equivalência será menor. Como poderá ser observado, à medida que os parâmetros utilizado pelo método de redução da complexidade forem alterados, o grau de equivalência também será alterado.

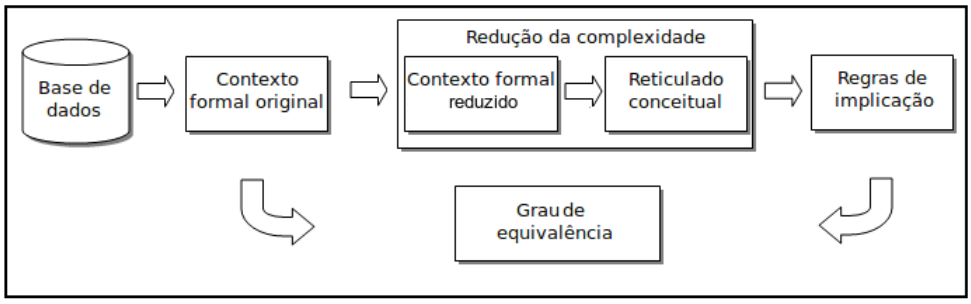

Figura 5.1: Grau de equivalência entre o contexto formal original e um conjunto de regras de implicação.

A questão chave nesse momento é: como mensurar o grau equivalência? Para essa questão são apresentadas duas métricas, a *fidelidade* e a *perda descritiva*.

Seja o contexto formal $K_o = (G_o, M_o, I_o)$ original, um contexto formal reduzido $K_r = (G_r, M_r, I_r)$, um reticulado conceitual $L_r = B(G_r, M_r, I_r)$ gerado a partir do contexto K_r e um conjunto reduzido à esquerda de regras $\varphi_r = \{f_1, f_2, ..., f_m\}$ extraído de L_r, em que cada regra f_i possui o formato $P \rightarrow Q$ ($P, Q \subseteq M_r$). Seguem as definições das métricas propostas.

A *fidelidade F* é definida como o índice de confiabilidade do conjunto de regras φ_r em relação ao conjunto de objetos originais. Se existe uma regra $f_i \in \varphi_r$, na qual $P \subset g'$ e $Q \not\subset g'$ para um objeto $g \in G_o$ então a regra f_i falhou, diminuindo a fidelidade. Na Equação 5.1 o cálculo da fidelidade é apresentado:

$$F = \frac{\sum_{i=1}^{|\varphi_r|} \left(1 - \frac{Nf_i}{|G_o|}\right)}{|\varphi_r|} \times 100 \tag{5.1}$$

em que F é a fidelidade em porcentagem, Nf_i é o número total de falhas da regra f_i, $|\varphi_r|$ é cardinalidade do conjunto de regras e $|G_o|$ é o número de objetos do contexto formal K_o original.

A *perda descritiva PD* é a perda da capacidade de caracterizar o conjunto de objetos originais G_o. Na Equação 5.2 o cálculo da perda descritiva é apresentado:

$$PD = (1 - \frac{\sum_{i=1}^{|G_o|} \left(\frac{|g'_{ri}|}{|g'_{oi}|}\right)}{|G_o|}) \times 100 \tag{5.2}$$

em que PD é a perda descritiva em porcentagem, $|g'_{ri}|$ é o número de atributos do objeto reduzido g_{ri} , $|g'_{oi}|$ é o número de atributos do objeto original g_{oi} e $|G_o|$ é o número de objetos do contexto formal K_o original.

Em relação às métricas propostas, a fidelidade mensura a possibilidade de sucesso das regras, sendo assim um índice que determina o quanto o conjunto de regras é confiável. Já a perda descritiva mensura a perda da capacidade de caracterizar o conjunto de objetos proporcionada pela perda de atributos.

Na próxima subseção o algoritmo para extração de regras de implicação será mostrado. Em seguida, na Seção 5.2, propostas para redução da complexidade do reticulado conceitual serão discutidas. Por último, no 6º capítulo as métricas de avaliação do conhecimento do reticulado conceitual serão aplicadas em situações reais.

5.1.1 Um Algoritmo para Extração de Regras de Implicação.

Como supracitado, pode-se extrair regras de implicação do reticulado conceitual. É possível extrair essas regras, que representam as informações contidas no contexto formal e consequentemente no reticulado conceitual, graças à hierarquia existente entre os conceitos.

Relembrando, uma implicação $P \rightarrow Q$ ocorre em um contexto formal (G,M,I) e consequentemente no reticulado conceitual L se, e somente se, $P' \subseteq Q'$. Desta forma, para que um algoritmo seja capaz de extrair todas as implicações de um reticulado conceitual, basta que para cada conceito formal (X,Y) se produza regras da forma $P \rightarrow Q$, em que $P \subseteq Y$, $Q = Y - P$ e que não exista nenhum conceito (W,Z) pai de (X,Y) tal que $P \subseteq Z$ [Carpineto e Romano, 2004a]. Nesse caso, P é chamado consistente com os pais do conceito formal (X,Y). Um algoritmo com essas características é o algoritmo *Find Implications* (Algoritmo 10) proposto por Carpineto et al. [1999]. Em Carpineto e Romano [2004a] e Vimieiro [2007] versões mais recente do algoritmo, no qual esse trabalho se baseia, também podem ser encontradas.

O algoritmo *Find Implications*, para cada conceito formal, exceto o ínfimo, chama a função *Find Implications From Concept*, Algoritmo 10 (linha 3), e salva as implicações resultantes. Cabe ressaltar, que o conceito ínfimo não é utilizado, pois ele pode gerar regras cujo consequente é vazio; apesar de válidas, essas regras não são relevantes.

Note que o procedimento descrito gera muitas regras redundantes. Para eliminar essa redundância, podem-se acrescentar restrições. Carpineto e Romano [2004a] propõem para o algoritmo *Find Implications From Concept* a extração de regras de implicação reduzidas a esquerda.

Para obter esse tipo de implicação, reduzida à esquerda, inicia-se a busca utilizando-se conjuntos de atributos mais gerais, percorrendo o reticulado em um processo de especialização dos atributos, enquanto o conjunto não se tornar consistente. Note que para dois conceitos formais (X,Y) e (W,Z) o primeiro será mais geral que o segundo se $Y \subset Z$. No Algoritmo 11 esse comportamento pode ser observado. Para a extração de implicações do conceito (X,Y), todos os seus pais são examinados (linha 3), já o teste de especialização é realizado para todos os conjuntos candidatos (variável *candSpecSet*) na linha 15.

Em relação à complexidade do algoritmo, segundo Carpineto e Romano [2004a] ela

Algoritmo 10 Algoritmo Find Implications

Entrada: Um reticulado conceitual $L = B(G,M,I)$
Saída: O conjunto de implicações φ

```
1: φ = ⊘
2: para cada (X,Y) ∈ {L - {( M′,M)}} faça
3:     φ = φ ∪ FindImplicationsFromConcept( (X,Y), L )
4: fim para
5: retorne φ
```

é $O(|C||K^2||M|q)$, em que $|C|$ é o número de conceitos formais do reticulado conceitual L, K é a maior cardinalidade do conjunto *lhsSet*, $|M|$ é o número de atributos e q é o maior número de pais por conceito.

Como exemplo, segue uma aplicação do algoritmo. Para o reticulado descrevendo as características dos animais apresentado na Figura 1.1, o qual possui 8 conceitos formais, o algoritmo *Find Implications* extraiu 12 regras, conforme pode-ser observado na Tabela 5.1.

Tabela 5.1: Conjunto de regras de implicação do reticulado conceitual para o exemplo dos animais.

#	**Regra**
1	{ *terrestre* } → { *resp. pulmonar* }
2	{ *resp. pulmonar* } → { *terrestre* }
3	{ *penas* } → { *terrestre, resp. pulmonar* }
4	{ *pêlos* } → { *terrestre, resp. pulmonar, glând. mamárias* }
5	{ *glând. mamárias* } → { *terrestre, resp. pulmonar, pêlos* }
6	{ *raciocínio* } → { *terrestre, resp. pulmonar, pêlos, glând. mamárias* }
7	{ *aquático* } → { *resp. branquial* }
8	{ *resp. branquial* } → { *aquático* }
9	{ *terrestre, aquático* } → { *resp. branquial, resp. pulmonar* }
10	{ *terrestre, resp. branquial* } → { *aquático, resp. pulmonar* }
11	{ *resp. pulmonar, aquático* } → { *terrestre, resp. branquial* }
12	{ *resp. pulmonar, resp. branquial* } → { *aquático, terrestre* }

Algoritmo 11 Função Find Implications From Concept

Entrada: Um reticulado conceitual $L = B(G,M,I)$ e um conceito formal $(X,Y) \in L$
Saída: Um conjunto de implicações S originadas do conceito formal (X,Y)

```
1:  S = ∅ //Conjunto de implicações do conceitos (X,Y)
2:  lhsSet = Y
3:  para cada (W,Z) pai de (X,Y) faça
4:     updLhsSet = ∅
5:     para cada lhs ∈ lhsSet faça
6:        se lhs ⊈ Z então
7:           updLhsSet = updLhsSet ∪ lhs
8:        senão
9:           candSpecSet = ∅
10:          para cada m ∈ Y − lhs faça
11:             candSpecSet = candSpecSet ∪ { lhs ∪ { m } }
12:          fim para
13:          para cada candS ∈ candSpecSet faça
14:             //Analisa a especialização dos elementos.
15:             se candS ⊈ Z E ∄ um elemento ⊆ candS em lhsSet - { lhs } então
16:                updLhsSet = updLhsSet ∪ candS
17:             fim se
18:          fim para
19:       fim se
20:    fim para
21:    lhsSet = updLhsSet
22: fim para
23: //Gera as implicações
24: para cada lhs ∈ lhsSet faça
25:    S = S ∪ { lhs → Y - lhs }
26: fim para
27: retorne S
```

5.2 Propostas de Redução do Reticulado Conceitual

Conforme apresentado no Capítulo 3, o conhecimento do usuário pode ser utilizado como parâmetro para redução da complexidade do reticulado conceitual; como exemplo, pode-se citar os métodos propostos por Belohlávek et al. [2004] e Stumme et al. [2002]. Esse conhecimento é útil por fornecer informações do mundo real, que não estão representadas no contexto formal, o que possibilita a agregação de conhecimento externo ao modelo. Esse conhecimento pode então ser utilizado para gerar apenas os conceitos formais considerados relevantes.

Nessa seção dois métodos para redução da complexidade do reticulado conceitual são propostos: o método de junção baseada em similaridade (Subseção 5.2.1) e o método de representação baseada em agrupamento (Subseção 5.2.2).

É importante ressaltar que os métodos de redução da complexidade do reticulado conceitual apresentados e propostos nessa seção consideram o *contexto formal limpo* (Subseção 3.8.1).

5.2.1 Junção Baseada em Similaridade - JBS

A primeira proposta de redução da complexidade do reticulado conceitual apresentada é chamada *junção baseada em similaridade* (JBS). Ela é aplicada no contexto formal e realiza um pré-processamento com o objetivo de associar/juntar objetos similares. Em outras palavras, um subconjunto de objetos similares, para um contexto formal, será representado por um único objeto.

A questão chave nesse momento é: como definir a similaridade entre objetos? Para atuar nessa questão propõe-se a criação de um peso para cada atributo do contexto formal (Definição 12), que representa sua relevância, e a partir desses valores determina-se a similaridade entre os objetos. Não é aplicado um peso diretamente em cada objeto, pois naturalmente a estrutura do reticulado conceitual classifica os objetos a partir dos atributos. Além disso, a relevância de um atributo influência a de vários objetos. Segue a definição da associação de peso aos atributos:

Definição 12. *Seja $M = \{m_1, m_2, ..., m_n\}$ um conjunto de atributos de um contexto formal. Um peso (relevância) w_i^m será um valor real na escala de 0 a 100 fornecido pelo usuário, que representa a relevância do atributo m_i.*

Note que a escala de valores imposta possui o objetivo de normalizar o processo.

Observe também que a relevância de cada atributo é definida pelo usuário. Ela deve ser baseada na carga semântica do atributo, ou seja, na importância do atributo para o modelo representado pelo contexto formal.

A partir da relevância definida em termos de um peso w_i^m para cada atributo $m_i \in M$, pode-se determinar a similaridade entre cada objeto. Essa medida é obtida através de uma matriz de *disparidade*, que representa o quanto dois objetos são distintos. A disparidade entre dois objetos $g_i, g_j \in G$, representada por $disp(i, j)$, é determinada por:

$$disp(i,j) = \frac{\sum_{k=1}^{|M|} \mu(i,j,k)}{\sum_{r=1}^{|M|} w_r^m} \times 100 \quad 1 \leqslant i < j \leqslant |G| \tag{5.3}$$

em que:

$$\mu(i,j,k) = \begin{cases} 0, & \text{se } g_i I m_k \leftrightarrow g_j I m_k \\ w_k^m, & \text{caso contrário.} \end{cases}$$

Em síntese a disparidade entre dois objetos $g_i, g_j \in G$ é determinada pela soma ponderada dos pesos w_i^m dos atributos que diferem. Note que quanto menor a disparidade entre dois objetos maior é sua similaridade.

A partir da matriz de disparidade pode-se determinar quais objetos são similares. Para isso, sugere-se a utilização de um índice de similaridade ε e um número máximo de elementos similares α. Considere dois objetos $g_i, g_j \in G$ similares se, e somente, se $disp(i, j) < \varepsilon$. No Algoritmo 12 um procedimento para determinar um conjunto de agrupamentos de objetos similares é apresentado. Note que a função auxiliar $ObjetoMenorDisparidade(disp, g_i, G)$ retorna o objeto com menor disparidade em relação ao objeto g_i dentre os objetos existentes em G; se houver mais de um, escolhe um qualquer. O conjunto de agrupamentos de objetos similares será representado por γ.

Note que o procedimento descrito pelo Algoritmo 12 cria grupos de objetos similares com no máximo α elementos. Determinado o conjunto de agrupamentos de objetos similares, pode-se agora formalizar o contexto formal reduzido (G_r, M_r, I_r).

Definição 13. *O contexto formal reduzido* (G_r, M_r, I_r) *será tal que* $G_r = \gamma$, $M_r = \bigcup\{\bigcap\{f' | f \in F\} | F \in \gamma\}$ *e* $(F, m) \in I_r$ *se, e somente se,* $m \in \bigcap\{f' | f \in F\}\}$.

Note que o objetivo do método é simplificar o contexto formal, representando objetos similares por um único objeto. Observe também que o impacto do método JBS será maior em processos de construção do reticulado conceitual que são mais sensíveis ao

Algoritmo 12 Função Agrupar Objetos

Entrada: Um conjunto de objetos G, uma matriz de disparidade $disp$, um número máximo de elementos similares α e um índice de similaridade ε.

Saída: O conjunto de agrupamentos de objetos γ.

```
1: γ = ⊘
2: enquanto G ≠ ⊘ faça
3:    g_i = menorObjetoEm(G).
4:    F = {g_i}
5:    G = G - {g_i}
6:    enquanto G ≠ ⊘ e |F| < α faça
7:       g_j = ObjetoMenorDisparidade (disp,g_i,G)
8:       se disp(i,j) ⩽ ε para todo i | g_i ∈ F então
9:          F = F ∪ {g_j}
10:         G = G - {g_j}
11:      senão
12:         saia
13:      fim se
14:   fim enquanto
15:   γ = γ ∪ {F}
16: fim enquanto
17: retorne γ
```

número de objetos.

Segue um exemplo de aplicação do método JBS para o contexto formal que representa a taxonomia dos animais, o qual é descrito na Tabela 1.1. Inicialmente é necessário propor um conjunto de pesos para os atributos do contexto formal. Na Tabela 5.2 o peso escolhido para cada atributo pode ser observado. Procurou-se aplicar um peso maior nas características mais visuais; entretanto, observe que a escolha desses valores é altamente subjetiva. Em seguida deve-se calcular a matriz de disparidade entre os objetos conforme proposto na Equação 5.3, como pode ser observado na Tabela 5.3.

Calculada a matriz de disparidade pode-se agrupar os objetos similares, conforme procedimento proposto no Algoritmo 12. Para o exemplo em discussão, utilizando um $\varepsilon = 5\%$ e $\alpha = 2$ objetos, o seguinte conjunto de objetos similares foi obtido: $\gamma\{\{1\},\{2\},\{3,4\},\{5\},\{6\}\}$. Observe que não existe mais distinção entre os objetos {3,4} (*humano* e *macaco*) e que o atributo {h} (*raciocínio*) foi removido. Após o processo de junção um novo contexto formal é obtido, como mostrado na Tabela 5.4.

Tabela 5.2: Peso aplicado ao objetos para o exemplo sobre taxonomia dos animais.

Objetos	**Atributos**								
	a	b	c	d	e	f	g	h	
	Pesos (w_i^m)								$\sum_{i=1}^{\|M\|} w_i^m$
	30	10	20	10	30	80	30	5	215
1		×		×					
2	×	×	×	×					
3		×		×	×		×	×	
4		×		×	×		×		
5		×		×		×			
6	×		×						

Tabela 5.3: Matriz de disparidade obtida para o exemplo sobre taxonomia dos animais.

	Objetos					
Objetos	**1**	**2**	**3**	**4**	**5**	**6**
1		23.2	30.2	27.9	37.2	32.5
2			53.4	51.1	60.4	9.3
3				2.3	67.4	62.7
4					65.1	60.4
5						69.7
6						

Tabela 5.4: Contexto formal para taxonomia de animais após aplicação do método JBS.

Objetos	**Atributos**						
	a	b	c	d	e	f	g
1		×		×			
2	×	×	×	×			
3 - 4		×		×	×		×
5		×		×		×	
6	×		×				

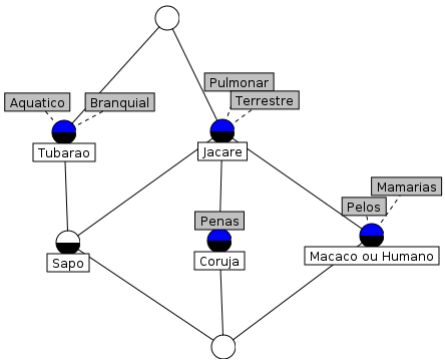

Figura 5.2: Diagrama de linhas após aplicação do método JBS.

O contexto formal reduzido apresentado na Tabela 5.4 gera um reticulado conceitual com 7 conceitos formais, que pode ser observado na Figura 5.2. Note que o número de conceitos formais teve um decréscimo de 1.

Em relação às métricas propostas para análise da qualidade do conhecimento representado, a redução proposta apresentou uma fidelidade de 100%, já a perda descritiva foi de 3.33%. Não houve decréscimo no índice de fidelidade para o exemplo em análise, pois o novo conjunto de regras é igual ao original da Tabela 5.1, excetuando-se a regra envolvendo o atributo *raciocínio* ($\{raciocínio\} \rightarrow \{$ *terrestre, resp. pulmonar, pêlos, glând. mamárias*$\}$). Contudo, a ausência do atributo *raciocínio* provoca uma perda descritiva para o objeto *humano*, refletindo na perda total citada.

Observe agora que o peso aplicado em cada atributo, o índice de similaridade ε e o número máximo de elementos α são os parâmetros que devem ser ajustados para obtenção de uma redução do contexto formal, e, consequentemente, do reticulado conceitual. Deve-se procurar um conjunto de parâmetros que propicie um número adequado de conceitos formais com grande similaridade e baixa perda descritiva. Note também que a escolha desses valores pode não ser uma tarefa trivial, e que ela deve ser baseada principalmente no conhecimento do usuário sobre a base em análise.

Note que caso fosse utilizado o mesmo peso para todos os atributos, a similaridade entre dois objetos seria determinada pelo número de atributos que os distinguisse. Em particular, para juntar os objetos que diferem um único atributo, basta fazer $\varepsilon = (1/8) \times 100 = 12.5\%$ e $\alpha = 2$, obtendo-se: $\gamma = \{\{2\},\{3,4\},\{1,5\},\{6\}\}$, e passaria a não haver distinção entre os objetos 3 e 4 (*humano* e *macaco*) e 1 e 5 (*jacaré* e *coruja*), conforme

pode ser observado na Tabela 5.5, que apresenta a matriz de disparidade para um peso igual. Entretanto, pode-se, por exemplo, considerar a característica possuir *pena* mais relevante do que os outros atributos por ser uma característica visual, fazer ajuste para cima em seu peso, e obter, assim, o mesmo resultado que foi obtido anteriormente.

Tabela 5.5: Matriz de disparidade obtida para o exemplo sobre taxonomia dos animais utilizando um peso igual a 1 em todos os atributos.

	Objetos					
Objetos	**1**	**2**	**3**	**4**	**5**	**6**
1		25	37.5	25	12.5	50
2			62.5	50	37.5	25
3				12.5	50	87.5
4					37.5	75
5						62.5
6						

Em relação aos grupos de técnicas para redução da complexidade do reticulado conceitual propostos na Seção 3.8, redução *baseada no conhecimento do usuário* e redução *baseada na eliminação de informação redundante* propostos, note que o método JBS é classificado com uma redução *baseada no conhecimento do usuário*. Já em relação aos principais métodos de redução da complexidade do reticulado conceitual observados na literatura, o método JBS é similar aos métodos que atuam no contexto formal (Subseção 3.8.1). Por exemplo, ao método apresentado por Wenxiu et al. [2005], que reduz o número de atributos do reticulado conceitual, e aos métodos apresentados por Gajdos et al. [2004] e Cheung e Vogel [2005], que utilizam técnicas do processo KDD para reduzir a dimensionalidade de massas de dados, nesse caso, do contexto formal.

O principal mérito do método JBS é aplicar uma forma consistente de redução do número de objetos, baseado no conhecimento do usuário. Em geral, a relevância dos atributos que descrevem um domínio é conhecida pelo usuário. E, caso não seja, pode-se descobri-la, por exemplo, através da análise de correlação, análise de componentes principais, etc. Esse conhecimento é, então, utilizado para reduzir o número de objetos, que em aplicações reais normalmente é muito superior ao número de atributos. Consequentemente, reduz-se o número de conceitos formais e o custo de se efetuar as derivações

5.2.2 Representação Baseada em Agrupamentos - RBA

A segunda proposta de redução da complexidade do reticulado conceitual é chamada *representação baseada em agrupamento* (RBA). Diferentemente do método JBS, que atua no contexto formal, o método RBA é aplicado durante o processo de construção do reticulado conceitual. A partir dos atributos de maior relevância, que são identificados pelo usuário para uma determinada aplicação, gera-se um lista de conceitos formais C. Essa lista é utilizada para construir o reticulado conceitual através de uma busca em largura, que inicia de cada conceito formal $c \in C$, até alcançar uma profundidade máxima, chamada de raio. Em síntese, o método RBA procura construir o reticulado conceitual considerando apenas os conceitos que orbitam os conceitos centrais, os quais são os primeiros conceitos formais que possuem os atributos considerados de maior relevância.

Antes de aplicar o método RBA é necessário determinar conjuntos de conjuntos de atributos $K \subseteq \mathcal{P}(M)$ considerados mais relevantes. Se $\{m_i\} \in K$, é porque se considera que os conceitos que contêm m_i em suas intensões não podem ser descartados. Por outro lado, se $\{m_i, m_j\} \in K$ são importantes os conceitos em que *ambos* ocorrem. E assim por diante. Observe que o conjunto K é o primeiro parâmetro de redução da complexidade utilizado pelo método RBA.

O segundo parâmetro necessário no método RBA é chamado raio r, que representa a partir de cada conceito formal $c \in C$ a maior profundidade e altura alcançáveis a partir de c. Note que o valor de r é o segundo parâmetro de redução da complexidade utilizado pelo método RBA. Uma segunda opção de parada, conforme poderá ser observado, é um número máximo de conceitos formais.

A partir dessa apresentação inicial pode-se descrever o método, conforme os passos apresentados a seguir:

1. determine o conceito supremo (G, G') e o conceito ínfimo (M', M) do contexto formal (G, M, I).

2. selecione os conjuntos de conjuntos $K \subseteq \wp(M)$ de atributos mais relevantes.

3. para cada conjunto $H \in K$ determine um conceito formal c, formando assim um conjunto C de conceitos formais, da seguinte forma: $C = \{(H', H'') \mid H \in K\}$.

4. faça uma expansão em largura partindo dos conceitos de C até alcançar o raio r, ou um número máximo de conceitos.

O método apresentado se concretiza no Algoritmo 13.

Algoritmo 13 Algoritmo RBA

Entrada: Um conjunto de conjuntos de atributos K, um contexto formal (G,M,I), um valor de raio r e um número máximo de conceitos n.
Saída: Um reticulado L.

```
1: i=0
2: C = { (H′,H″) | H ∈ K}
3: inicialize L com os conceitos formais (G,G′) e (M′,M)
4: adicione em L cada conceito c ∈ C
5: sup = c
6: inf = c
7: enquanto i ⩽ r e L possui menos de n conceitos faça
8:    S = ⋃ { vizinhos superiores de c | c ∈ sup }
9:    I = ⋃ { vizinhos inferiores de c | c ∈ inf }
10:   sup = S
11:   inf = I
12:   adicione em L cada conceito c ∈ sup
13:   adicione em L cada conceito c ∈ inf
14:   i=i+1
15: fim enquanto
16: retorne L
```

A seguir um exemplo de aplicação do método RBA é apresentado. Seja o contexto formal para taxonomia dos animais apresentado na Tabela 1.1 e o reticulado conceitual equivalente apresentado na Figura 1.1. Suponha que no universo descrito pelo contexto formal, os atributos (h) *raciocínio* e (c) *respiração branquial* apresentem maior importância para a aplicação que se tenha em mente, ou seja, $k = \{\{h\},\{c\}\}$. Considere também um raio $r = 2$ e $n = 2^{|M|}$ (nesse caso o número de conceitos é irrelevante).

Nesse caso, a lista de conceitos C de maior representatividade será $\{(\{3\}\{b,d,e,g,h\}),(\{2,6\}\{a,c\})\}$ que representam (3) *humano* possuindo as características (b) *terrestre*, (d) *respiração pulmonar*, (e) *pêlos*, (g) *glândulas mamárias* e (h) *raciocínio* e (2) *sapo* e (6) *tubarão* possuindo as características (a) *aquático* e (c) *respiração branquial*.

Na Figura 5.3 os conceitos formais iniciais gerados são apresentados; observe que nesse caso tem-se um raio igual a 1 ($r = 1$). Na Figura 5.4 a estrutura do reticulado para $r = 2$ é apresentada, e finalmente, na Figura 5.5 o reticulado obtido para $r = 3$ pode ser observado. Note que o reticulado final obtido não é idêntico ao reticulado original apresentado na Figura 1.1. O conceito formal $(\{5\}, \{b, d, f\})$, que representa o animal (5) *coruja* possuindo as características (b) *terrestre*, (d) *respiração pulmonar* e (d) *penas* não é gerado pelo método. Essa diferença é provocada pelo processo de construção do reticulado conceitual proposto pelo método RBA.

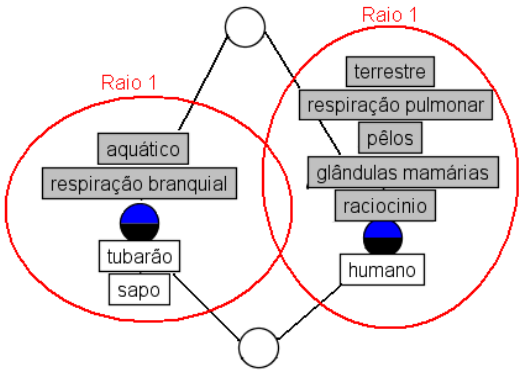

Figura 5.3: Reticulado conceitual criado pelo método RBA para *r*=1.

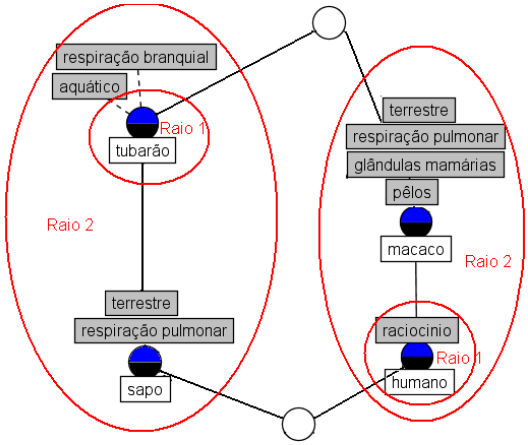

Figura 5.4: Reticulado conceitual criado pelo método RBA para *r*=2.

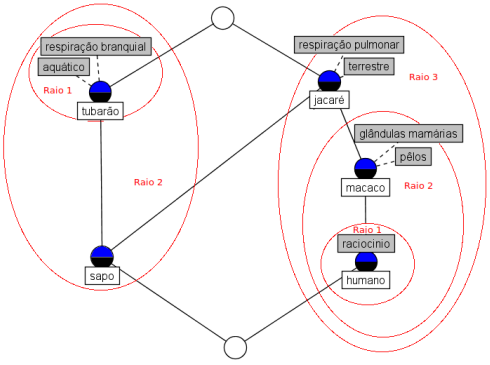

Figura 5.5: Reticulado conceitual criado pelo método RBA para *r*=3.

Note também que alguns valores de raio podem provocar inconsistências no reticulado. Por exemplo, na Figura 5.4 os conceitos formais $(\{2\},\{a,b,c,d\})$ e $(\{4\},\{b,d,e,g\})$ que representam o animal (2) *sapo* possuindo as características (a) *aquático*, (b) *terrestre*, (c) *respiração branquial* e (d) *respiração pulmonar* e o animal (4) *macaco* possuindo as características (b) *terrestre*, (d) *respiração pulmonar*, (e) *pêlos* e (g) *glândulas mamárias* respectivamente, compartilham as características (b) *terrestre* e (d) *respiração pulmonar*. Entretanto, para $r = 2$ não existe um conceito superior que possui somente as características (b) e (d). Observe que apenas para $r = 3$, Figura 5.5, esse conceito é adicionado. Novamente, essa diferença é provocada pelo processo de construção do reticulado conceitual proposto pelo método RBA, o qual realiza uma expansão em largura partindo dos conceitos chaves. Note que não são criado conceitos formais, e sim identificados aqueles pertencentes ao reticulado conceitual original, que estão dentro de um parâmetro estipulado.

Em relação às métricas propostas para análise da qualidade do conhecimento representado, na Tabela 5.6 a fidelidade e a perda descritiva para cada valor de raio *r* podem ser observadas. Note que, com o aumento do valor de raio, a fidelidade aumenta e a perda descritiva diminui. Ou seja, a perda torna-se menor e o conhecimento representado mais consistente.

Tabela 5.6: Fidelidade e Perda descritiva para o contexto formal descrevendo a taxonomia dos animais.

Raio r	**Fidelidade (%)**	**Perda Descritiva (%)**
1	76.19	58.33
2	74.07	33.33
3	85.18	13.88

Em relação aos grupos de técnicas para redução da complexidade do reticulado conceitual propostos na Seção 3.8, redução *baseada no conhecimento do usuário* e redução *baseada na eliminação de informação redundante* propostos, note que o método RBA é classificado com uma redução *baseada no conhecimento do usuário*. Já em relação aos principais métodos de redução da complexidade do reticulado conceitual observados na literatura, o método RBA é similar aos métodos que atuam no reticulado conceitual (Subseção 3.8.2). Por exemplo, ao método apresentado por Arévalo et al. [2007], que propõe a geração de sub-hierarquias conceituais, e ao método apresentado por Stumme

et al. [2002], que propõe apenas a geração de conceitos formais frequentes para construção do reticulado conceitual.

O principal mérito do método RBA é selecionar do reticulado conceitual apenas o conjunto de conceitos formais mais relevantes, e em seguida, construir apenas a hierarquia conceitual próxima desses conceitos. Criando dessa forma, um reticulado conceitual reduzido.

Capítulo 6

Análise dos Resultados

Este capítulo tem como objetivo avaliar os principais paradigmas de construção dos reticulados conceituais através do *benchmark* proposto na Seção 4.2. Também serão avaliadas as técnicas de redução da complexidade propostas na Seção 5.2 utilizando o processo de avaliação discutido na Seção 5.1.

Todos os testes realizados foram feitos utilizando o arcabouço *EF-Concept Analysis* (Seção 4.1) em um computador com as seguintes características:

- fabricante: Dell;
- modelo: Latitude D630;
- processador: Intel(R) Core (TM) 2 Duo T7100 - 1.80 GHz, 1.80 GHz;
- memória RAM: 2 GB;
- tipo de sistema: sistema operacional de 32 bits;
- sistema operacional: Ubuntu 9.04;
- máquina virtual Java: versão 1.6.0-11.

Cabe ressaltar que não é possível ter controle sobre possíveis interferências da linguagem de programação utilizada e que todos os testes foram realizados utilizando apenas memória primária.

6.1 Avaliações do Benchmark Proposto

Nas subseções a seguir é feita uma avaliação dos paradigmas de construção do reticulado conceitual para o *benchmark* real e sintético propostos.

6.1.1 Avaliações do Benchmark Sintético

Inicialmente, sugeriu-se a avaliação dos algoritmos/paradigmas para o pior caso (Seção 3.1), o qual possui apenas a diagonal principal não preenchida. Na Tabela 6.1 o desempenho para esse caso pode ser observado[1].

Conforme comentado, para o pior caso, o número de conceitos formais tem um comportamento exponencial. Na última coluna da Tabela 6.1 esse comportamento pode ser observado. Note que, a partir de $|G| = |M| = 18$ apenas o algoritmo *Next Closure* foi capaz de gerar os conceitos formais. Na literatura esse algoritmo é citado como um dos melhores algoritmos existentes, principalmente para contextos formais densos. Os resultados observados na Tabela 6.1 confirmam esse fato. Note também que para $|G| = |M| = 20$ o algoritmo *Next Closure* gastou quase duas horas (01:46:26:825) para gerar os conceitos formais.

Como o algoritmo *Next Closure* apenas fornece os conceitos formais é necessário utilizar o algoritmo *Concept Cover* para ordenar os conceitos, formando assim o reticulado conceitual. Note que nesse caso o custo total dos dois algoritmos foi muito alto, sendo a maior parte desse custo proveniente do algoritmo *Concept Cover*.

Ainda analisando os resultados da Tabela 6.1, pode-se concluir também que contextos formais com $|G| > 18$, $|M| > 18$ e uma alta densidade, para o arcabouço *EF-Concept Analysis*, necessitam do uso de memória secundária. Note também que, apesar de utilizar uma medida em milissegundos, para contextos formais com $|M| = |G| = 2$, 3 e 4 houve uma discrepância nas medidas. Por exemplo, para os algoritmos *Next Neighbours* e *Godin* com $|M| = |G| = 3$ tem-se um tempo de 0 ms. A partir de $|M| = |G| = 5$ os valores mensurados ficam estáveis.

O segundo critério indicado para análise no *benchmark* é a densidade. Inicialmente, assumiu-se uma distribuição aleatória da densidade em todo contexto formal da se-

[1] Foram utilizadas a notações: "*" para indicar que a memória principal disponível não foi suficiente para gerar os conceitos formais e hora (hh), minuto (mm), segundo (ss) e milissegundos (ms) para indicar o tempo gasto.

Tabela 6.1: Desempenho dos paradigmas de construção do reticulado conceitual para o pior caso.

	hh:mm:ss:ms					
Objetos e Atributos	**Next Closure**	**Concept Cover**	**Next Closure + Concept Cover**	**Next Neighbours**	**Godin**	**Nº de Conceitos Formais**
2	00:00:00:001	00:00:00:004	00:00:00:005	00:00:00:002	00:00:00:002	4
3	00:00:00:001	00:00:00:001	00:00:00:002	00:00:00:000	00:00:00:000	8
4	00:00:00:001	00:00:00:002	00:00:00:003	00:00:00:001	00:00:00:001	16
5	00:00:00:002	00:00:00:003	00:00:00:005	00:00:00:004	00:00:00:002	32
6	00:00:00:002	00:00:00:003	00:00:00:005	00:00:00:005	00:00:00:004	64
7	00:00:00:003	00:00:00:006	00:00:00:009	00:00:00:008	00:00:00:005	128
8	00:00:00:006	00:00:00:021	00:00:00:027	00:00:00:011	00:00:00:010	256
9	00:00:00:007	00:00:00:083	00:00:00:090	00:00:00:022	00:00:00:034	512
10	00:00:00:018	00:00:00:336	00:00:00:354	00:00:00:053	00:00:00:122	1024
11	00:00:00:049	00:00:01:171	00:00:01:220	00:00:00:094	00:00:00:410	2048
12	00:00:00:140	00:00:05:450	00:00:05:590	00:00:00:213	00:00:01:677	4096
13	00:00:00:455	00:00:13:012	00:00:13:467	00:00:00:526	00:00:07:194	8192
14	00:00:01:748	00:00:47:180	00:00:48:928	00:00:01:472	00:00:28:068	16384
15	00:00:06:128	00:03:52:738	00:03:58:866	00:00:03:794	00:01:50:554	32768
16	00:00:25:451	00:17:54:059	00:18:19:510	00:00:11:019	00:08:13:023	65536
17	00:01:50:947	*	*	00:00:31:160	00:31:10:268	131072
18	00:06:37:575	*	*	*	*	262144
19	00:26:26:298	*	*	*	*	524288
20	01:46:26:825	*	*	*	*	1048576

guinte forma: $I = 5, 10, ..., 95$ (%). Em seguida, assumiu-se os mesmos valores para uma distribuição aleatória com mesmo número de atributos por objeto. Para cada valor de densidade utilizou-se ($|G| = 20$ e $|M| = 50$) e ($|G| = 50$ e $|M| = 20$). Note que, conforme proposto na definição do *benchmark*, a inversão da quantidade de objetos e atributos (quantidade maior de atributos e em seguida uma quantidade maior de objetos) possui o objetivo de detectar se o algoritmo em análise é mais sensível ao número de objetos ou ao número de atributos. Nas Figuras 6.1, 6.2, 6.3 e 6.4 os gráficos do comportamento para esses casos, em função do número de conceitos formais e do tempo em milissegundos, podem ser observados.

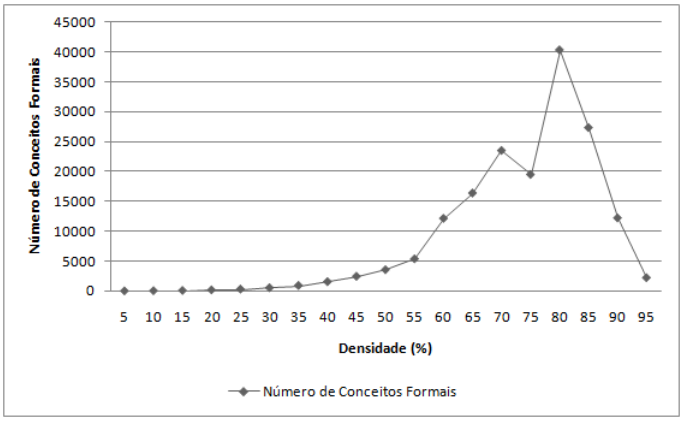

Figura 6.1: Número de conceitos para $|G| = 20$, $|M| = 50$ e uma distribuição aleatória.

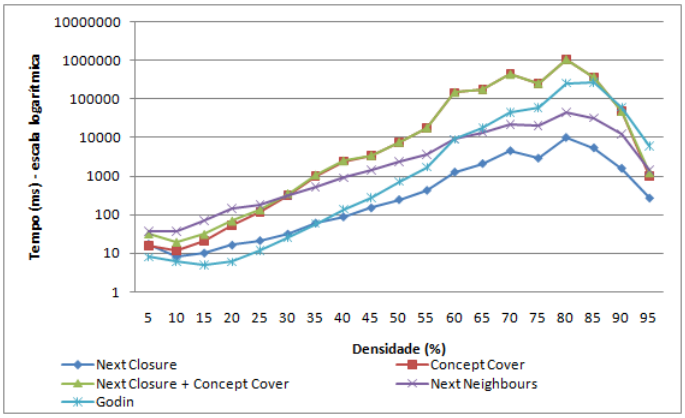

Figura 6.2: Tempo em milissegundos para $|G| = 20$, $|M| = 50$ e uma distribuição aleatória.

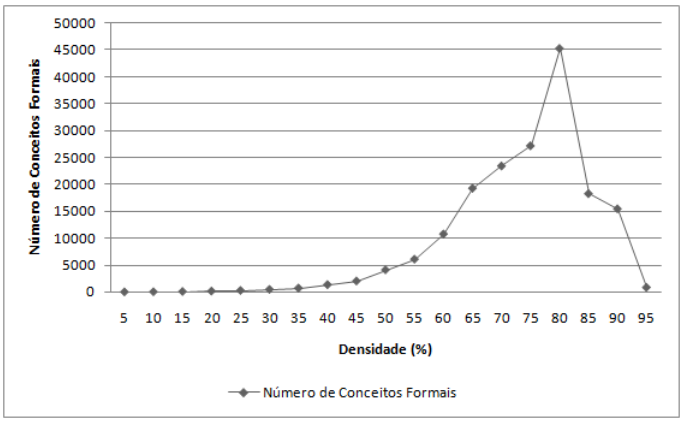

Figura 6.3: Número de conceitos para $|G| = 50$, $|M| = 20$ e uma distribuição aleatória.

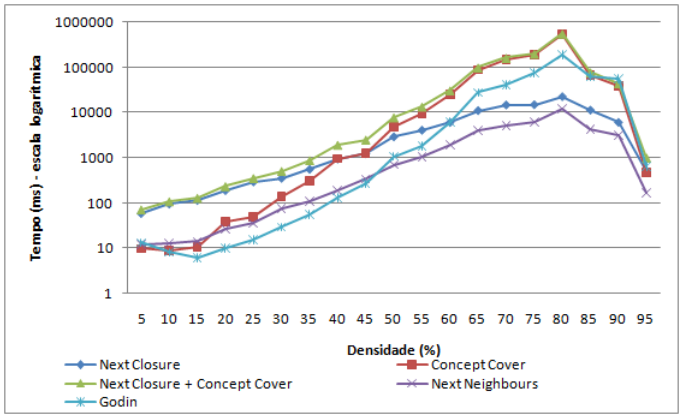

Figura 6.4: Tempo em milissegundos dos paradigmas para $|G| = 50$, $|M| = 20$ e uma distribuição aleatória.

Inicialmente, observe que os gráficos apresentados nas Figuras 6.1 e 6.3, que descrevem o número de conceitos em função da variação da densidade, apresentam um comportamento semelhante.

Em relação ao desempenho dos algoritmos, para uma distribuição aleatória da densidade acima de 40%, com uma quantidade maior de atributos o desempenho do algoritmo *Next Closure* foi melhor, conforme pode ser observado no gráfico da Figura 6.2. Ao inverter a quantidade de objetos e atributos, gráfico da Figura 6.4, o desempenho do algoritmo *Next Closure* sofreu uma alteração, ficando o melhor desempenho com o algoritmo *Next Neighbours*. Observe também que o pior desempenho foi para a soma

do tempo gasto pelos algoritmos *Next Closure* e *Concept Cover*, já que nesse caso todos os conceitos são gerados e posteriormente ordenados no reticulado conceitual. Note também que o maior tempo é gasto no processo de ordenação dos conceitos formais pelo algoritmo *Concept Cover*.

Ainda em relação ao tempo gasto pelos algoritmos, note que o algoritmo mais indicado para uma densidade alta é o algoritmo *Next Neighbours*. Observe também que para valores de densidade abaixo de 35% o algoritmo *Godin* é mais indicado. À medida que a densidade aumenta, até aproximadamente 80%, o desempenho do algoritmo *Godin* torna-se pior, pois a cada novo objeto inserido é necessário atualizar uma grande quantidade de conceitos formais.

Outra importante observação no conjunto de gráficos 6.1, 6.2, 6.3 e 6.4 é em relação ao decréscimo do número de conceitos formais e do tempo para geração dos conceitos, após uma densidade de aproximadamente 80%. Como supracitado, assumiu-se uma distribuição aleatória da densidade em todo contexto formal. Nesse caso, à medida que a densidade aumenta, começam a surgir objetos que possuem todos os atributos e atributos que se aplicam a todos os objetos. Note que esse comportamento tende a reduzir o número de conceitos formais, já que menos conceitos são necessários para criar a hierarquia conceitual.

A sequência de gráficos apresentada nas Figuras 6.5, 6.6, 6.7 e 6.8 realiza o mesmo teste descrito anteriormente, variando-se a densidade para ($|G| = 20$ e $|M| = 50$) e ($|G| = 50$ e $|M| = 20$). Contudo, com uma densidade média fixa por objeto (cada objeto $g \in G$ possui a mesma quantidade de atributos). Note que nesse caso, não é observado um decréscimo no número de conceitos formais, conforme ocorreu para distribuição aleatória da densidade para todo contexto formal. Isso ocorre pois, nesse caso, não são gerados objetos nos quais todos os atributos se aplicam.

Em relação ao desempenho, conforme apresentado nos gráficos 6.6 e 6.8, observa-se um comportamento similar à da distribuição da densidade de forma aleatória em todo contexto formal (Figuras 6.2 e 6.4). Isto é, melhor desempenho para os algoritmos *Next Closure* e *Next Neighbours* para um alta densidade; contudo, apenas o algoritmo *Next Closure* foi capaz de chegar a uma densidade de 95%.

Após a análise do comportamento dos algoritmos para diversos valores de densidade, sugere-se uma análise em relação ao número de objetos. Conforme discutido na Seção 4.2.1, sugere-se a variação do número de objetos, para uma quantidade

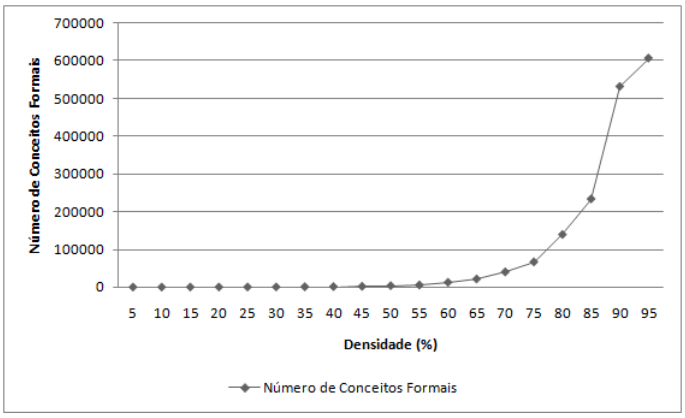

Figura 6.5: Número de conceitos para $|G| = 20$, $|M| = 50$ e uma densidade média.

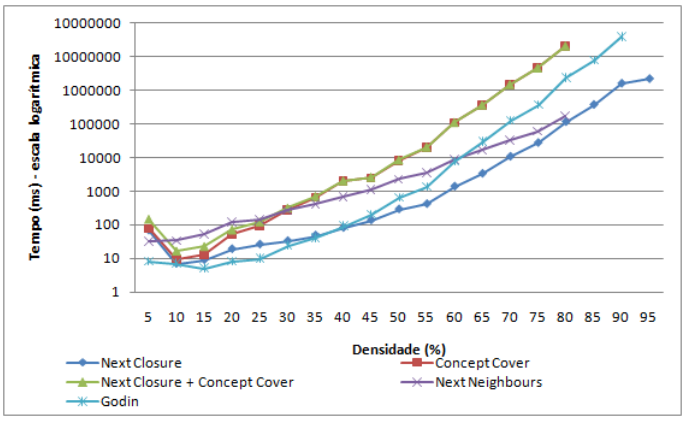

Figura 6.6: Tempo em milissegundos dos paradigmas para $|G| = 20$, $|M| = 50$ e uma densidade média.

fixa de atributos e uma distribuição da densidade aleatória também fixa. Nos gráficos apresentados nas Figuras 6.9 e 6.10 o comportamento para essa análise, variando $|G| = 20, 40, ..., 200$, utilizando $|M| = 20$ e uma distribuição aleatória da densidade correspondente a 50% de todo contexto formal ($|I| = 50\%$), podem ser observados.

Inicialmente, observa-se que o crescimento do número de conceitos é menor para uma variação do número de objetos, conforme mostra a Figura 6.9, quando comparada com a variação da densidade nos gráficos das Figuras 6.1, 6.3, 6.5 e 6.7, o que confirma as observações encontradas na literatura[Carpineto e Romano, 2004a]. Observe também que, novamente, o desempenho do algoritmo *Next Neighbours* foi melhor e que o desempenho do algoritmo *Next Closure*, juntamente com algoritmo *Concept Cover*, foi

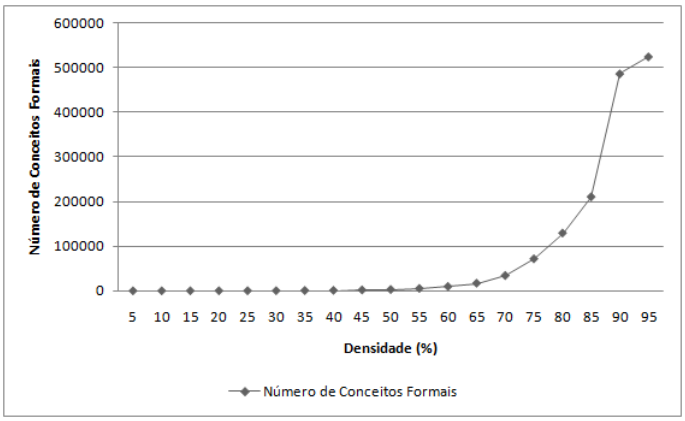

Figura 6.7: Número de conceitos para $|G| = 50$, $|M| = 20$ e uma densidade média.

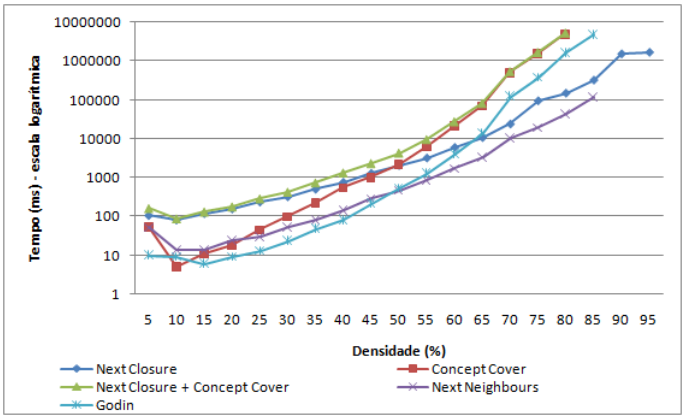

Figura 6.8: Tempo em milissegundos dos paradigmas para $|G| = 50$, $|M| = 20$ e uma densidade média.

pior.

Já nas Figuras 6.11 e 6.12 o último grupo de testes proposto, variar o número de atributos, para uma quantidade fixa de objetos e uma distribuição aleatória da densidade também fixa, podem ser observados. Utilizou-se uma variação do número de atributos $|M| = 5, 10, ..., 100$, uma quantidade fixa de objetos ($|G| = 20$) e uma distribuição aleatória da densidade $|I| = 50\%$. Em problemas do mundo real esse comportamento é raramente encontrado, já que o domínio da aplicação (número de objetos) já é conhecido. Entretanto, o teste descrito é relevante por demonstrar o comportamento dos algoritmos nessa configuração. Caso o algoritmo em análise apresente melhores resul-

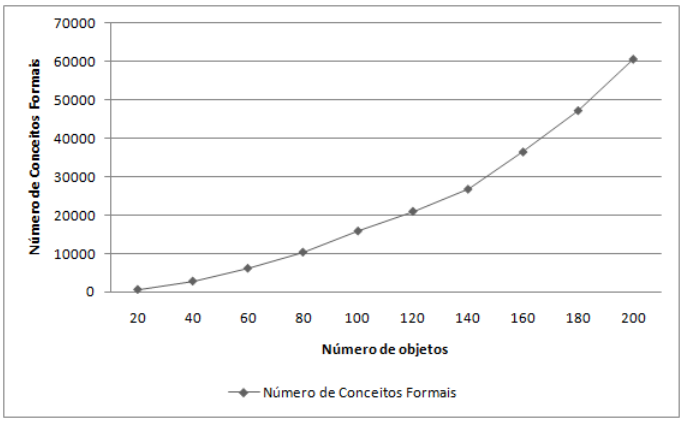

Figura 6.9: Número de conceitos para $|M| = 20$, $|I| = 50\%$ aleatória e uma variação do número de objetos.

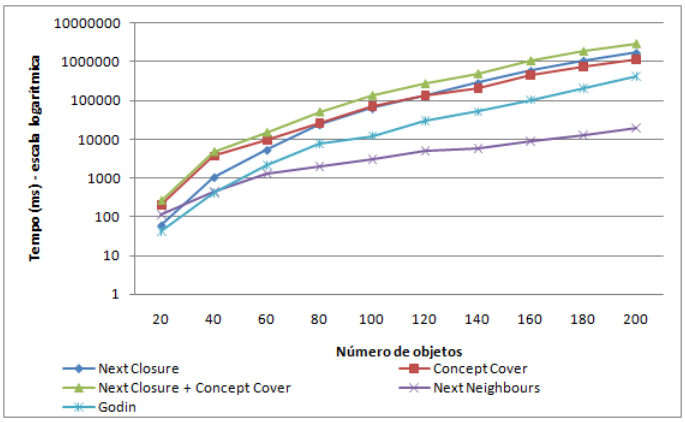

Figura 6.10: Tempo em milissegundos dos paradigmas para $|M| = 20$, $|I| = 50\%$ aleatória e uma variação do número de objetos.

tados pode-se, por exemplo, trabalhar com um contexto formal dual (invertendo objetos e atributos).

Observe novamente que o crescimento do número de conceitos formais, Figura 6.11, foi menor quando comparada com o crescimento ocorrido durante uma variação da densidade, observado nas Figuras 6.1, 6.3, 6.5 e 6.7.

Observe também que, ao variar o número de atributos, o algoritmo *Next Closure* teve um melhor desempenho. Contudo, o custo do algoritmo *Next Closure* acrescido do tempo do algoritmo *Concept Cover*, novamente, foi muito elevado. Nas condições

em análise é mais adequado a utilização do algoritmo *Godin*. Note que nesse caso o algoritmo de *Godin* foi melhor pois o número de objetos não é elevado.

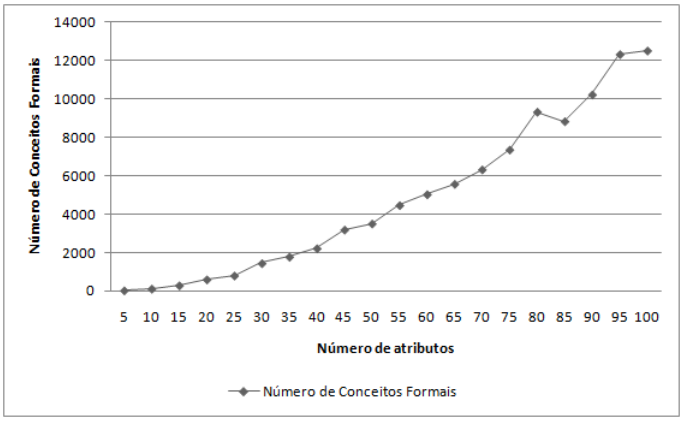

Figura 6.11: Número de conceitos para $|G| = 20$, $|I| = 50\%$ aleatória e uma variação do número de atributos.

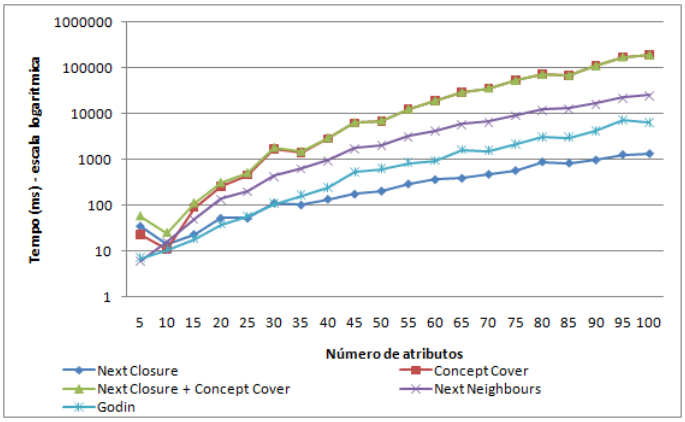

Figura 6.12: Tempo em milissegundos dos paradigmas para $|G| = 20$, $|I| = 50\%$ aleatória e uma variação do número de atributos.

6.1.2 Avaliações do Benchmark Real

Como parte do *benchmark* proposto para avaliação dos algoritmos e paradigmas de construção do reticulado conceitual propõe-se a aplicação dos algoritmos em bases de dados reais. Na Tabela 6.2 o desempenho dos paradigmas de construção do reticulado conceitual para o conjunto de bases de dados reais apresentadas como parte do

benchmark proposto pode ser analisado. Considerando as características dos contextos formais reduzidos apresentados na Tabela 4.5, seguem alguns comentários:

Para o contexto formal Water nenhum algoritmo foi capaz de obter os conceitos formais após oito horas de execução; note que, esse contexto formal possui mais de 8 milhões de conceitos formais[2], conforme pode ser observado na última linha da Tabela 6.2. Quando se compara a base de dados *Water* com problemas do mundo real de alta dimensionalidade fica visível a necessidade de técnicas para redução da complexidade dos reticulados conceituais, a proposta de novas abordagens para geração dos conceitos, etc. Mesmo possuindo um contexto formal com apenas 316 objetos, 76 atributos e uma densidade de 51.18%, o número conceitos formais é elevado, o que inviabilizou a utilização dos algoritmos.

Tabela 6.2: Aplicações do *benchmark* real nos paradigmas de construção do reticulado conceitual.

	hh:mm:ss:ms				
Contextos Formais	**Next Closure**	**Concept Cover**	**Next Closure + Concept Cover**	**Next Neighbours**	**Godin**
Íris	00:00:00:071	00:00:00:052	00:00:00:123	00:00:00:051	00:00:00:052
Wine	00:01:10:727	00:09:35:070	00:10:45:797	00:00:06:054	00:00:21:272
Yeast	00:00:03:159	00:00:00:716	00:00:03:875	00:00:00:258	00:00:00:422
Water	*	*	*	*	*

Ainda na Tabela 6.2, para a base de dados Yeast, o maior tempo foi gasto pelo algoritmo *Next Closure* acrescido do tempo gasto pelo algoritmo *Concept Cover*. Entretanto, ao contrário do comportamento observado para os dados sintéticos, o maior custo foi do algoritmo *Next Closure*. Esse comportamento deve-se ao fato do contexto formal gerado para a base de dados Yeast apresentar uma baixa densidade, e consequentemente, poucos conceitos formais. Isso torna mais custoso as operações de derivação realizadas pelo algoritmo *Next Closure*, que o processo de ordenação dos conceitos formais realizado pelo algoritmo *Concept Cover*. Essas constatações indicam que o algoritmo *Next Closure* não é indicado para contextos formais com uma densidade muito baixa. Uma solução algorítmica para diminuir esse custo é manter o contexto formal ordenado pelo número de incidências em cada objeto, e uma copia do contexto formal ordenado pelo

[2]O número total de conceitos formais foi obtido utilizando o software Conexp[Yevtushenko, 2000].

número de incidências em cada atributo, semelhante à proposta apresentada por Fu e Nguifo [2003].

Para o contexto formal representando a base de dados Wine, o melhor desempenho foi observado pelo algoritmo *Next Neighbours*. Note que o contexto formal para essa base de dados possui uma quantidade de objetos muito superior à quantidade de atributos ($|G| = 132$ e $|M| = 29$). Por último, para o contexto formal produzido para a base de dados Íris, todos os algoritmos apresentaram um comportamento muito similar. Novamente, o pior tempo foi do algoritmo *Next Closure* acrescido do tempo gasto pelo algoritmo *Concept Cover* e o melhor desempenho foi do algoritmo *Next Neighbours*.

Conforme citado, um algoritmo projetado para ter alto desempenho é capaz de superar facilmente alguns dos contexto formais apresentados nesta seção, sejam eles reais ou sintéticos. Entretanto, cabe ressaltar que o objetivo aqui é demonstrar o processo de análise dos algoritmos, isto é, os procedimentos que devem ser adotados para análise dos algoritmos de construção dos reticulados conceituais. Ao *benchmark* proposto pode-se adicionar novos conceitos formais, formando assim um *benchmark* mais amplo, capaz de avaliar algoritmos com maior eficiência.

6.2 Avaliação dos Métodos de Redução da Complexidade

Nas subseções a seguir é feita uma avaliação do método de *junção baseada em similaridade* (Seção 5.2.1) e do método de *representação baseada em agrupamentos* (Seção 5.2.2) através da metodologia proposta na Seção 5.1. Os testes foram feitos utilizando as bases de dados reais propostas na Seção 4.2.2.

6.2.1 Avaliação do Método de Junção Baseada em Similaridade

Conforme discutido na Subseção 5.2.1 o método de *junção baseada em similaridade* (JBS) necessita de um conjunto de pesos que representam a importância de cada atributo do contexto formal. Para determinar esse conjunto de pesos é necessário um estudo das bases de dados em análise ou o auxílio de um especialista no domínio da aplicação, o qual seja capaz de identificar a relevância de cada atributo. Esse estudo foge ao escopo desse trabalho, o qual pretende apenas demonstrar que a partir de um conjunto de pesos pode-se reduzir o contexto formal e consequentemente o reticulado conceitual.

Contudo, note que a abstração do nível da aplicação também é útil para esse trabalho. Principalmente, por apresentar resultados independentes da aplicação, evitando assim, por exemplo, resultados tendenciosos ou muito específicos para uma determinada aplicação, como frequentemente é observado na literatura.

Partindo desse princípio, propôs-se um conjunto de pesos para cada base de dados real (Íris, Wine, Yeast e Water) baseado na frequência dos atributos. Foi adotada a seguinte premissa: *quanto maior a frequência, maior será a relevância e consequentemente o peso do atributo*. Note que os contextos formais que representam as bases de dados reais são contextos formais limpos, conforme discutido na Seção 3.8.1 (não existem objetos que possuem os mesmos atributos ou atributos possuídos pelos mesmos objetos). Além disso, não existem objetos comuns a todos os atributos ou atributos comuns a todos os objetos.

Ainda assim, o leitor mais atento pode questionar sobre a abordagem empregada na determinação da relevância dos atributos. Contudo, cabe ressaltar ainda que quanto mais frequente é um atributo, maior é sua presença nos conceitos formais, isto é, quanto maior a frequência de um atributo maior sua influência nos conceitos formais.

Determinado o peso de cada atributo, o método JBS necessita de um parâmetro α que determina o número máximo de objetos que podem ser considerados similares. Utilizou-se para cada base de dados o valor máximo de 20% do número de objetos, o que corresponde a $\alpha = 4, 26, 22$ e 63 para as bases Íris, Wine, Yeat e Water, respectivamente. Já o parâmetro ε foi variado entre 0% e 100% cada base de dados em análise.

Estabelecidos os parâmetros necessários para utilização do método JBS pode-se analisar seus resultados. Nas Figuras 6.13 e 6.14 os resultados para a base de dados Íris podem ser observados. É necessário ressaltar que todos os gráficos apresentados possuem valores em porcentagem em relação ao contexto formal original (sem aplicação dos métodos de redução da complexidade). Inicialmente, no gráfico apresentado na Figura 6.13 observa-se o número de objetos $|G|$, a densidade $|I|$, o número de conceitos formais e o número de regras para variações de ε. Observe que com o aumento do valor de ε o número de objetos $|G|$, a densidade $|I|$, o número de conceitos formais e o número de regras tendem a diminuir, isso ocorre pois mais objetos são considerados similares. Note também que existem regiões para as quais o aumento do parâmetro ε não influencia os parâmetros em análise.

Já o gráfico da Figura 6.14 apresenta a fidelidade média (fidelidade do conjunto de

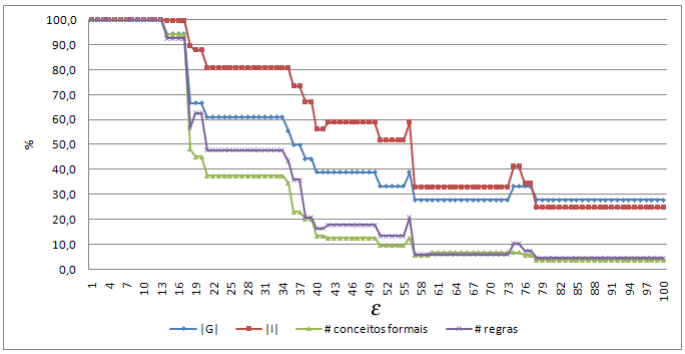

Figura 6.13: Número de objetos, densidade, número de conceitos formais e número de regras obtidos para a base Íris aplicada no método JBS.

regras, Seção 5.1) e a perda descritiva para variações do parâmetro ε. Como pode ser observado no gráfico, conforme os valores de ε aumentam, a fidelidade média tende a diminuir. Por outro lado, a perda descritiva tende a aumentar, já que uma quantidade maior de objetos são eliminados.

Analisando os gráficos das Figuras 6.13 e 6.14 em conjunto nota-se primeiramente que para valores de $\varepsilon \leqslant 13\%$ não é obtido nenhuma redução, e consequentemente os índices de fidelidade e perda descritiva são máximos. Isso ocorre quando não existem objetos $g_i, g_j \in G$ similares (disp(i,j) $\leqslant \varepsilon$). Observe que a primeira redução obtida foi para $\varepsilon = 14\%$, que apresentou 94.4% do número de conceitos formais, uma fidelidade de 99.7% e uma perda descritiva de 2.8%. Caso uma redução de aproximadamente 6% no número de conceitos formais seja considerada satisfatória (torne possível a geração dos conceitos formais, obtenção do reticulado conceitual e a análise do reticulado), pode-se considerar uma redução ótima já que as perdas obtidas são mínimas. Por outro lado, caso seja necessário uma redução ainda maior, um $\varepsilon = 40\%$ pode ser considerado relevante, já que nesse caso tem-se 56% dos conceitos formais (redução de 44%), uma fidelidade de 88% e uma perda descritiva de 44.9%. Nesse caso, note que mesmo com uma perda descrita alta a qualidade do conhecimento ainda é relevante.

Em relação aos resultados para base de dados Íris, os valores que originaram os gráficos das Figuras 6.13 e 6.14 podem ser observados no Anexo B nas Tabelas B.1 e B.2. Nessas tabelas também pode-se analisar a fidelidade mínima é máxima, a fidelidade média e o desvio padrão. A fidelidade mínima e máxima são valores de fidelidade por regra, a fidelidade média é a fidelidade do conjunto de regra, conforme discutido na

Seção 5.1, e o desvio padrão é a dispersão dos dados em relação a fidelidade média. Como pode ser observado no anexo, a fidelidade máxima sofreu uma alteração para valores de $\varepsilon \geqslant 56\%$, demonstrando que algumas regras sempre falharam; já a fidelidade mínima sofreu um decréscimo para valores de $\varepsilon \geqslant 14\%$.

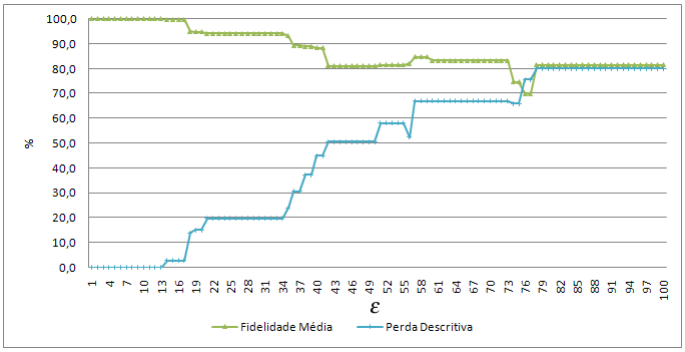

Figura 6.14: Fidelidade média e perda descritiva para a base de dados Íris aplicada no método JBS.

Para a segunda base de dados em análise, base Wine, seus resultados podem ser observados nos gráficos apresentados nas Figuras 6.15 e 6.16. Inicialmente, no gráfico da Figura 6.15 observe que para $\varepsilon = 8\%$ o número de objetos foi reduzido para 77% e o número de conceitos formais para aproximadamente 70%; a base de dados Wine originalmente apresenta um total de 29032 conceitos formais. Note que para esse mesmo valor de ε a fidelidade foi de aproximadamente 99% e a perda descritiva de aproximadamente 3.8%, conforme pode ser observado no gráfico da Figura 6.16. Observe que $\varepsilon = 8\%$ pode ser considerado um valor bom, por apresentar uma redução significativa aliada com uma alta representatividade e uma perda descritiva baixa. Observe ainda que um segundo valor de ε que pode ser considerado adequado, caso $\varepsilon = 8\%$ não possa ser considerado viável, é $\varepsilon = 15\%$. Nesse caso tem-se 18.46% dos conceitos formais, uma fidelidade de 97% e uma perda descritiva de aproximadamente 18.8%. Note que apesar da grande redução, os índices de qualidade mensurados continuam elevados.

Note que pode ser interessante utilizar um $\varepsilon = 15\%$ mesmo sendo viável trabalhar com $\varepsilon = 8\%$, pois o primeiro valor de ε apresenta uma redução maior e índices de fidelidade e perda descritiva relevantes.

Ainda no gráfico da Figura 6.15 observe que a alteração do número de objetos $|G|$ é mais suave, quando comparada com alteração da densidade $|I|$. Isso ocorre pois, os ob-

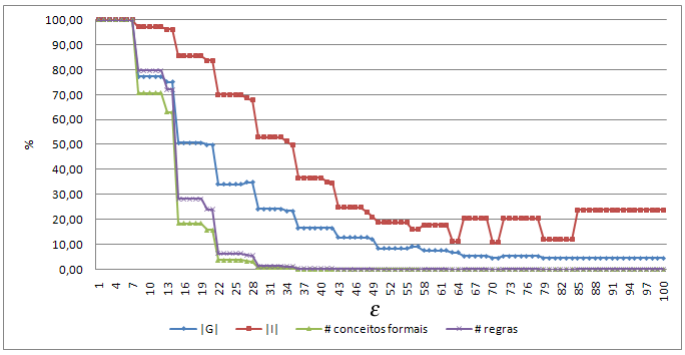

Figura 6.15: Número de objetos, densidade, número de conceitos formais e número de regras obtidos para a base Wine aplicada no método JBS.

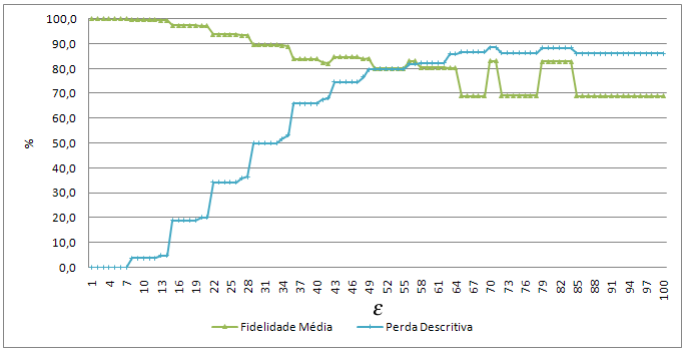

Figura 6.16: Fidelidade média e perda descritiva para a base de dados Wine aplicada no método JBS.

jetos de um conjunto $F \in \gamma$ de objetos similares (ver Seção 5.2.1) podem apresentar uma quantidade similar de atributos, provocando uma alteração pequena na densidade. Note também que para $\varepsilon \geqslant 85\%$ não houveram mais reduções nos parâmetros em análise. No gráfico da Figura 6.16, que apresenta as métricas para avaliação do conhecimento representado para a base de dados Wine, esse mesmo comportamento pode ser observado. Os resultados que proporcionaram os gráficos apresentados nas Figuras 6.15 e 6.16 podem ser observados no Anexo B nas Tabelas B.3 e B.4.

A terceira base de dados em análise é a base Yeast, seus resultados podem-se observados nos gráficos apresentados nas Figuras 6.17 e 6.18, que apresentam as reduções (número de objetos $|G|$, a densidade $|I|$, o número de conceitos formais e o número de regras) e as métricas (fidelidade e perda descritiva) respectivamente

Inicialmente, observe na Figura 6.17 que para $\varepsilon = 1\%$ o número de objeto $|G|$ foi reduzido para aproximadamente 98%, já o número de conceitos foi reduzido para aproximadamente 99%. Para esse mesmo valor de ε, conforme pode-ser observado no gráfico da Figura 6.18, a fidelidade foi de aproximadamente 99% e perda descritiva de aproximadamente 0.4%. Note que uma primeira redução mais significante ocorre para $\varepsilon = 3\%$ que apresenta 85% dos conceitos formais, uma fidelidade de 99% e uma perda descritiva de 4%.

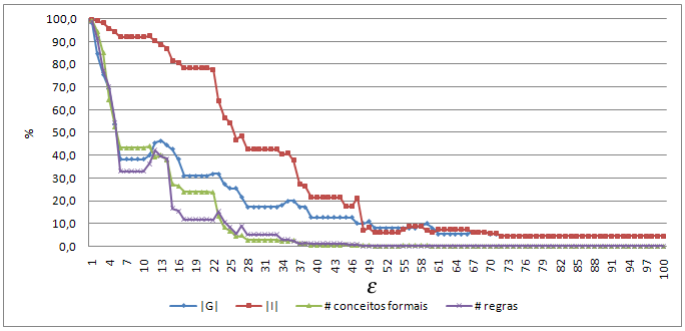

Figura 6.17: Número de objetos, densidade, número de conceitos formais e número de regras obtidos para a base Yeast aplicada no método JBS.

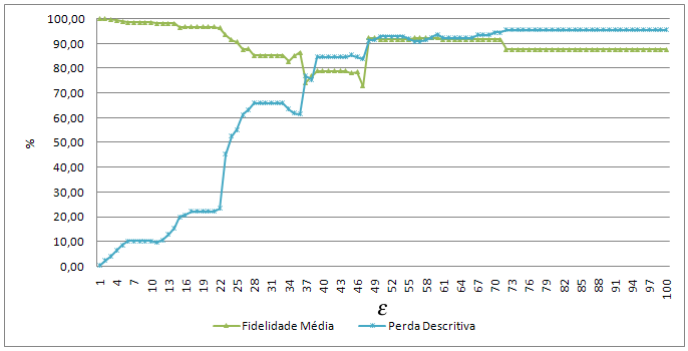

Figura 6.18: Fidelidade média e perda descritiva para a base de dados Yeast aplicada no método JBS.

Note que outro valor interessante é $\varepsilon = 6\%$ que apresenta aproximadamente 38% dos conceitos formais, uma fidelidade de 98% e uma perda descritiva de 10%. Observe que apesar da grande redução, ainda assim o contexto formal apresenta uma boa capacidade descritiva e uma ótima fidelidade.

Observe ainda analisando o gráfico da Figura 6.17 que em alguns casos tem-se uma redução muito rápida dos parâmetros em análise, por exemplo, como ocorrido para a variação de ε entre 1% e 7% para o número de objetos, conceitos e regras. Isso ocorre pois o parâmetro ε é alterado sempre 1%. Nesse caso, para obter uma análise mais pontual pode-se variar ε mais gradualmente, por exemplo, 0.1%.

Os resultados que proporcionaram os gráficos apresentados nas Figuras 6.17 e 6.18 podem ser observados no Anexo B nas Tabelas B.5 e B.6.

A última base de dados em análise é a base Water, seus resultados podem ser analisados nos gráficos das Figuras 6.19 e 6.20. Inicialmente, observe que não foi possível gerar os conceitos formais para $\varepsilon < 14\%$. Além disso, não foi possível obter o número de regras para o contexto original (o contexto original apresenta 8940648 de conceitos formais). Dessa forma, o gráfico apresentado na Figura 6.19 não possui a porcentagem de regras obtidas para cada valor de ε. Note que para contextos formais que apresentem uma grande quantidade de conceitos formais pode ser inviável analisar sua fidelidade. Entretanto, ao aplicar a redução esse cálculo torna-se viável.

Observe na Figuras 6.19 que para o primeiro valor de ε alcançado, $\varepsilon = 14\%$, o número de objetos foi de aproximadamente 31% do contexto formal e o número de conceitos formais de aproximadamente 2.14% do contexto formal original (191801 conceitos formais). Esse mesmo valor de ε apresentou uma fidelidade de aproximadamente 95% e uma perda descritiva de aproximadamente 21%. Note que mesmo com uma perda descritiva média e uma grande redução no número de conceitos formais a fidelidade do conhecimento representado foi de 95%. Observe ainda que valores de $\varepsilon > 58\%$ não é observado nenhuma redução.

Note que para a base de dados Water uma grande redução foi necessária para que fosse possível gerar os conceitos formais. Para valores de $\varepsilon < 14\%$ não foi possível obter os conceitos e gerar o reticulado conceitual, isso demonstra que valores de $\varepsilon < 14\%$ produzem uma quantidade de conceitos formais inviável para o arcabouço *EF-Concept Analysis*. Os resultados que proporcionaram os gráficos apresentados nas Figuras 6.19 e 6.20 podem ser observados no Anexo B nas Tabelas B.7 e B.8.

Como observado, através do método JBS foi possível reduzir os contextos formais das bases de dados em análise e consequentemente o reticulado conceitual. Por exemplo, para a base dados Water, a qual não foi possível obter o reticulado conceitual para o contexto formal original, utilizando um parâmetro de corte adequado, foi possível

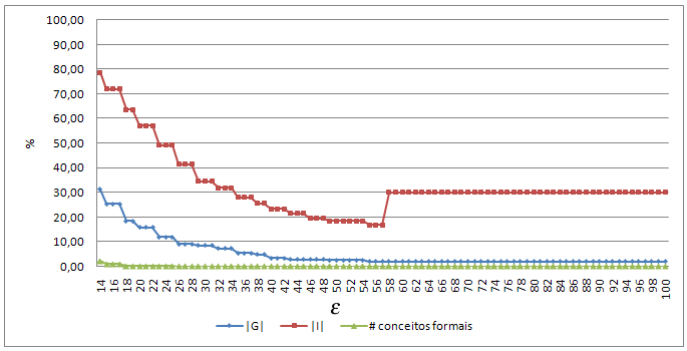

Figura 6.19: Número de objetos, densidade e número de conceitos formais para a base Water aplicada no método JBS.

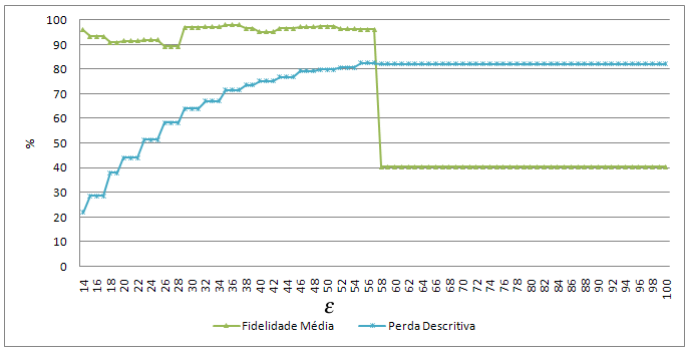

Figura 6.20: Fidelidade média e perda descritiva para a base de dados Water aplicada no método JBS.

construir o reticulado conceitual. Note também que foi possível controlar a qualidade do conhecimento representado no reticulado, demonstrando assim a potencialidade do método JBS.

Observe ainda que a escolha dos parâmetros de ajuste do método JBS não é uma tarefa fácil. Primeiramente, deve-se procurar escolher um conjunto de pesos que realmente represente a relevância de cada atributo. Em seguida, determinar o número máximo de objetos que podem ser considerados similares e por último determinar o melhor valor de ε. Note que os parâmetros mais relevantes são o peso dos atributos e o valor de ε escolhido. Deve-se procurar escolher um ε que torne viável a geração dos conceitos formais, a construção do reticulado conceitual e a análise do reticulado. Caso o valor de ε escolhido apresente índices de fidelidade e perda descritiva considerados inaceitáveis, deve-se procurar alterar os demais parâmetros de ajuste do método JBS

(pesos e número máximo de objetos similares) para se obter índices melhores.

Em relação as métricas utilizadas para análise do conhecimento, fidelidade e perda descritiva, note que ambas foram capazes de descrever a qualidade da informação existente no reticulado conceitual. Observe também que o cálculo da perda descritiva pode ser feito facilmente analisando o contexto formal. Por outro lado, a análise da fidelidade pode nem sempre ser considerada viável, principalmente para bases de dados de alta dimensionalidade. Isso deve-se as características do conjunto de regras extraído do reticulado conceitual. Note que para cada conceito formal podem existir mais de uma regra. Entretanto, ainda assim a fidelidade é relevante, por ser uma primeira proposta na literatura de análise independente da qualidade do conhecimento existente no reticulado conceitual. Note ainda que, a medida que o contexto formal é reduzido a análise da fidelidade torna-se viável, já que uma quantidade menor de conceitos formais é obtida.

6.2.2 Avaliação do Método de Representação Baseada em Agrupamentos

Conforme discutido na Subseção 5.2.2 o método de *representação baseada em agrupamento* (RBA) necessita de um conjunto de conjuntos de atributos $K \subseteq \mathcal{P}(M)$ considerados mais relevantes. Novamente, a escolha desses parâmetros foge ao escopo desse trabalho. Assim, adotou-se o seguinte procedimento: *para as bases de dados Íris, Wine e Yeast escolheu-se atributos do contexto formal que representam o parâmetro classe da base de dados original. Já para a base de dados Water escolheu-se três atributos do contexto formal de forma aleatória*. Cabe ressaltar que cada atributo considerado de maior relevância para as quatro bases em análise formou um conjunto unitário em K.

O segundo parâmetro necessário no método RBA é o número de conceitos formais n. Nesse caso, adotou-se $n = 2^{|M|}$ para cada base de dados em análise. Note que nesse caso o número de conceitos é irrelevante.

Definidos os parâmetros necessários para o método RBA, pode-se analisar os resultados para cada base dados. Na Tabela 6.3 os resultados obtidos para a base de dados Íris podem ser analisados. A primeira linha da tabela representa os valores originais, já as demais representam os valores em porcentagem em relação ao valor original. Note que a fidelidade mínima e máxima são valores de fidelidade por regra, a fidelidade média é a fidelidade do conjunto de regras, conforme discutido na Seção 5.1, já o desvio padrão representa dispersão dos dados em relação a fidelidade média.

Tabela 6.3: Base Íris aplicada no método RBA.

			Fidelidade				
Raio	**# Conceitos Formais (%)**	**# Regras (%)**	**Mín. (%)**	**Máx. (%)**	**Méd. (%)**	**Desv. Pad. (%)**	**Perda Desc. (%)**
	103	**67**	**100**	**100**	**100**	**0**	**0**
1	4.85	2.99	55.55	100	77.77	31.42	75
2	14.56	14.93	61.11	100	83.88	12.4	62.22
3	32.04	41.79	38.88	100	81.34	16.91	53.05
4	49.51	68.66	38.88	100	84.54	14.76	49.72
5	64.08	91.04	38.88	100	85.51	13.61	47.5

Ainda na Tabela 6.3 note que a medida que o raio aumenta, o número de conceitos formais e consequentemente de regras tende a aumentar. Além disso, a fidelidade média melhora e a perda descritiva tende a diminuir. Note que pelo processo de construção do reticulado conceitual mesmo utilizando um raio r e um número de conceitos formais infinito, ainda assim, podem existir perdas de fidelidade e representatividade, o que deve-se ao processo de construção do reticulado conceitual.

Observe também na Tabela 6.3 que para um raio $r = 5$ o número de conceitos formais foi de 64.08%, já fidelidade corresponde foi de 85.51% e a perda descritiva de 47.5%. Para $r \geqslant 6$ não houveram alterações no número de conceitos formais, demonstrando assim, que o supremo e o ínfimo haviam sido alcançados em todas as ramificações do processo de construção do reticulado conceitual. Nesse caso, se os valores de fidelidade e perda descritiva não sejam aceitáveis deve-se procurar alterar o conjunto de conjuntos K de atributos mais importantes, adicionando novos conjuntos de atributos, até alcançar índices que possam ser considerados satisfatórios. Note também que foram escolhidos apenas os conceitos formais que orbitam os conceitos formais que representam os atributos classes. Dessa forma, por exemplo, para um processo de classificação, apenas os conceitos centrais foram selecionados.

Já na Tabela 6.4 pode-se observar os resultados para a base de dados Wine. Note que para um raio $r = 6$ tem-se 17.42% dos conceitos formais, uma fidelidade de 89.70% e uma perda descritiva de 76.08%. Note novamente que apesar da alta perda descritiva, os dados representados no reticulado conceitual continuam consistentes, já que a fidelidade desses dados é de 89.70%, conforme pode ser observado na última linha da tabela.

Ainda na Tabela 6.4 note que para $r > 3$ não existem alterações na perda descritiva. Isso ocorre quando o processo de construção do reticulado conceitual do método JBS não insere novos conceitos formais que descrevem os objetos[3]. Nesse caso, novamente se a perda descritiva resultante não for adequada deve-se adicionar novos conjuntos, ao conjunto K (ver Subseção 5.2.2).

Tabela 6.4: Base Wine aplicada no método RBA.

			Fidelidade				
Raio	**# Conceitos**	**# Regras**	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
	Formais (%)	**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
	29031	**40645**	**0**	**100**	**100**	**0**	**0**
1	0.02	0.01	69.69	100	83.33	13.62	86.2
2	0.17	0.12	48.48	100	73.35	10.49	80.68
3	1.01	0.72	46.21	100	84	10.59	76.08
4	3.61	2.65	37.87	100	88.42	11.43	76.08
5	9.09	7.09	37.12	100	89.61	12.61	76.08
6	17.42	14.35	38.63	100	89.70	13.84	76.08

Para a terceira base de dados em análise, base Yeast, seus resultados podem ser observados na Tabela 6.5. Para $r = 8$ tem-se 63.19% dos conceitos formais, uma fidelidade de 77.75% e uma perda descritiva de 51.81%, conforme pode ser observado na última linha da tabela. Novamente, como ocorrido na base de dados Wine (Tabela 6.4) para $r > 3$ não houveram alterações na perda descritiva.

Por último, na Tabela 6.6 os resultados para a base de dados Water podem ser observados. Observe que novamente não foi possível obter o conjunto de regras para o contexto formal original, que apresenta um total de 8940648 conceitos formais. Note que para $r = 6$ o número de conceitos formais foi de 0.1391% do contexto original (que corresponde a 12435 conceitos formais). Já a fidelidade corresponde foi de 61.96% e a perda descritiva de 86.9%, conforme pode ser observado na última linha da Tabela 6.6. Para valores de $r > 6$ não foi possível gerar o reticulado conceitual.

Note que caso o número total de conceitos formais não seja considerado satisfatório ou os valores mensurados para fidelidade e perda descritiva, deve-se procurar adicionar novos atributos ao conjunto K de atributos de maior relevância. Observe também que

[3]Um objeto g com g' atributos é descrito por um conceito formal c se, e somente se, $c = \{\{g\}, \{g'\}\}$.

Tabela 6.5: Base Yeast aplicada no método RBA.

			Fidelidade				
Raio	**# Conceitos**	**# Regras**	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
	Formais (%)	**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
	1929	**1199**	**100**	**100**	**100**	**0**	**0**
1	0.62	3.67	9.99	100	40.88	33.76	67.77
2	3.47	7.34	9.99	100	51.73	24.73	55.55
3	10.68	22.6	9.99	100	57.9	20.66	51.81
4	23.07	41.95	9.99	100	63.36	19.5	51.81
5	37.74	61.13	9.99	100	69.27	18.32	51.81
6	50.96	79.98	9.99	100	73.59	17.24	51.81
7	59.2	92.74	9.99	100	75.88	17.06	51.81
8	63.19	99.17	9.99	100	77.75	17.03	51.81

Tabela 6.6: Base Water aplicada no método RBA.

			Fidelidade				
Raio	**# Conceitos**	**# Regras**	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
	Formais (%)	**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
	8940648	**-**	**100**	**100**	**100**	**0**	**0**
1	0.0001	2	50.31	100	75.15	35.13	97.5
2	0.0005	42	49.68	100	58.34	13.68	96.22
3	0.0030	351	19.3	100	58.99	16.86	92.85
4	0.0133	1886	19.62	100	60.07	18.97	92.2
5	0.0472	7545	10.44	100	61.52	19.97	91.6
6	0.1391	7546	9.81	100	61.96	20.39	86.9

o número máximo de conceitos alcançáveis está altamente relacionado com os recursos computacionais disponíveis. Por exemplo, utilizando na base de dados Water todos os atributos do contexto formal no conjunto K de atributos, formando conjuntos unitários, para o método JBS obtém-se para $r = 6$ um total de 114089 conceitos formais. Para valores de $r \geqslant 6$ a memória principal disponível não foi suficiente para gerar os conceitos formais e a estrutura do reticulado conceitual.

Como pode ser observado, em geral a perda descritiva provocada pelo método JBS é muito alta, principalmente quando comparada com o método RBA. Isso deve-se ao processo de construção do reticulado conceitual, o qual mantém seu foco apenas nos conceitos formais que orbitam os conceitos formais centrais (conceitos que representam

os atributos considerados mais relevantes). Nesse caso, sendo necessário obter melhores índices de perda descritiva deve-se adicionar mais conjuntos de atributos ao conjunto K. Por outro lado, a fidelidade, em geral, apresenta valores altos, demonstrando assim a qualidade ainda aceitável do conhecimento representado no reticulado conceitual.

Outra característica observada em todas as bases de dados é que sempre existiram regras que falharam. Essa característica deve-se novamente ao processo de construção do reticulado conceitual, o qual sempre elimina conceitos formais.

Note também que a escolha de um conjunto de parâmetros ótimos para o método RBA é diferente, principalmente quando comparado com o processo empregado no método JBS. O foco do método RBA é a seleção de um conjunto central de conceitos formais (aqueles que orbitam conceitos considerados de maior relevância). Desta forma, o valor de raio r escolhido deve apenas representar um nível de hierarquia considerada satisfatória em vez de focar-se na redução total obtida. Em outras palavra, o foco do método RBA é a seleção de um pequeno grupo de conceitos formais considerados chaves para análise de um processo. Por outro lado, o método JBS procura extrair um número máximo de conceitos formais viáveis para análise do reticulado conceitual que apresente valores considerados ótimos de fidelidade e perda descritiva.

Capítulo 7
Conclusões

Esse capítulo tem como objetivo sumarizar as contribuições desse trabalho e discutir os possíveis trabalhos futuros.

Os principais algoritmos para geração dos conceitos formais e construção do reticulado conceitual foram discutidos: *Next Closure*, *Concepts Cover*, *Next Neighbours* e *Godin*. Observou-se que o algoritmo *Next Closure* é o indicado para construção dos conceitos formais quando não se deseja gerar o reticulado conceitual. O algoritmo *Concepts Cover* é o indicado para o processo de ordenação dos conceitos formais, quando se deseja construir o reticulado conceitual a partir de um conjunto de conceitos formais. O algoritmo *Next Neighbours* é o indicado para o processo de geração dos conceitos formais e construção do reticulado conceitual em um único processo. Por último, o algoritmo *Godin* é o indicado como representante do paradigma incremental, quando deseja-se construir o reticulado conceitual inserindo novos objetos em um contexto formal. Na Tabela 7.1 um pequeno resumo cronológica de todos os algoritmos citados nesse trabalho pode ser observado.

Foi apresentado um arcabouço didático chamado *EF - Concept Analysis* (*Educational Framework for Concept Analysis*). Proposta esta inspirada na necessidade do desenvolvimento de algoritmos para análise formal de formal ágil e prática, sem a necessidade de trabalhar com massas de dados (contextos formais) de alta dimensionalidade.

Também foi apresentado um *benchmark* para avaliação dos métodos de construção do reticulado conceitual. Esse *benchmark* foi inspirado naqueles apresentados na literatura e nas características dos algoritmos apresentados. O *benchmark* proposto procurou explorar os algoritmos para o pior caso, e para variações da densidade, número de objetos e atributos. Além disso, os dados reais utilizados pelo *benchmark* passaram por um

Tabela 7.1: Resumo cronológico dos principais algoritmos.

Nome/autor	**Gera o reticulado**	**Comentários**
Malgrange [1962]	Não	Primeiro algoritmo, inicialmente utilizado para outras aplicações.
Chein [1969]	Não	Melhoria do trabalho apresentado por Malgrange [1962].
Fay [1973]	Não	Pouco referenciado/utilizado na literatura.
Norris [1978]	Não	Versão incremental de uma primeira versão do algoritmo.
Next Closure [Ganter, 1984]	Não	Principal algoritmo conhecido na literatura.
Bordat [1986]	Sim	Primeiro algoritmo capaz de gerar os conceitos formais e o reticulado conceitual.
Godin et al. [1991]	Sim	Primeiro algoritmo capaz de atualizar o reticulado conceitual.
Carpineto e Romano [1993]	Sim	Muito aplicado em recuperação de informação
Nourine e Raynaud [1999]	Sim	Algoritmo de uso mais genérico.
Concepts Cover [Nourine e Raynaud, 1999]	Sim	Algoritmo eficiente para ordenar os conceitos formais no reticulado conceitual.
Valtchev e Missaoui [2002]	Sim	Proposta de construção do reticulado conceitual baseada no paradigma de divisão e conquista.
Qiao et al. [2003]	Sim	Adaptação do algoritmo de Nourine capaz de atualizar o reticulado.
Next Neighbours [Carpineto e Romano, 2004a]	Sim	Considerado um evolução do algoritmo *Bordat*.

processo de adequação (modelagem para um contexto formal) bem detalhado, o qual pode ser facilmente utilizado em outros trabalhos. Frequentemente é observado na literatura pouca descrição sobre as bases de dados utilizadas para análise dos algoritmos ou métodos de redução da complexidade do reticulado conceitual. Por esse motivo, procurou-se detalhar ao máximo o processo de construção do *benchmark* proposto.

Foi apresentado uma proposta para avaliação das técnicas de redução da complexidade do reticulado conceitual. Raramente é encontrado na literatura abordagens para comparar tais técnicas. Feita essa constatação, esse trabalho procurou apresentar uma proposta independente da aplicação do reticulado conceitual ou da técnica utilizada para reduzir sua complexidade. Essa proposta foi fundamentada em duas métricas: a *fide-*

lidade e a *perda descritiva*. A primeira métrica é calculada a partir de um conjunto de regras que são extraídas do reticulado conceitual, que representam o conhecimento existente no contexto formal e consequentemente no reticulado conceitual, mensurando assim a consistência do conhecimento. Já a perda descritiva mensura a perda da capacidade de representar o modelo descrito no contexto formal original.

Foram apresentadas também duas propostas para redução da complexidade do reticulado conceitual. A primeira chamada *junção baseada em similaridade* (JBS), que atua no contexto formal a partir um de conjunto de pesos que representam a importância dos atributos de um contexto formal. A segunda proposta chamada *representação baseada em agrupamento* (RBA), que atua no processo de construção do contexto formal, selecionando apenas os conceitos formais principais que orbitam conceitos formais que representam os atributos considerados relevantes pelo especialista no domínio da aplicação.

É interessante que os métodos para redução da complexidade dos reticulados conceituais apresentem certas características, descritas a seguir:

- 1º - evitar criar objetos, atributos ou incidências;
- 2º - procurar preservar ao máximo a hierarquia conceitual original;
- 3º - apresentar baixas perdas descritivas;
- 4º - apresentar uma alta fidelidade (o conhecimento do reticulado conceitual reduzido dever ser consistente com o do original).

A criação de objetos, atributos ou incidências altera o modelo descrito pelo contexto formal e, consequentemente, pelo reticulado conceitual; assim, tais alterações devem ser evitadas. Entre os conceitos formais existem relacionamentos, os quais possuem informação importante, portanto deve-se procurar mantê-los. O contexto formal representa um modelo do mundo real, o qual também é representado pelo reticulado conceitual. Dessa forma, os métodos de redução da complexidade devem apresentar baixas perdas descritivas (perda da capacidade de representar o contexto formal). Por fim, o conhecimento representado no reticulado conceitual deve ser consistente com o original, dessa forma, deve-se procurar os melhores índices de fidelidade.

Os métodos JBS e RBA não inserem objetos, atributos ou incidências. Pelo contrário, as duas propostas de redução da complexidade do reticulado conceitual procuram valorizar aspectos relevantes do modelo descrito pelo contexto formal. Note também que as duas propostas procuram destacar os relacionamentos mais relevantes entre os conceitos formais. O método RBA, por exemplo, apenas seleciona os conceitos formais que orbitam conceitos formais centrais, selecionando dessa forma apenas as relações mais importantes. Por fim, os dois métodos apresentam parâmetros que permitem ajustar a perda descritiva e a fidelidade para os índices considerados satisfatórios para cada aplicação.

Na aplicação do *benchmark* aos algoritmos estudados, chegou-se a conclusão que quando for necessário apenas gerar os conceitos formais, o algoritmo *Next Closure* é altamente recomendado. Como citado na literatura, e constatado nesse trabalho, o algoritmo apresenta um bom desempenho, principalmente para contextos formais com grande densidade. Por outro lado, quando for necessário obter o reticulado conceitual, o desempenho total pode não ser satisfatório. Apesar do algoritmo *Concept Cover* ser citado na literatura como o principal algoritmo para ordenação dos conceitos formais, ele é pouco indicado já que o custo para ordenação dos conceitos formais mostrou-se muito elevado.

Considerando-se a necessidade de gerar os conceitos e o reticulado conceitual tem-se:

- pior caso: poucos conceitos formais (até $|G| = |M| = 16$) o algoritmo *Godin* apresenta bons resultados, para contextos maiores o algoritmo *Next Neighbours* é mais indicado;
- contexto formais com poucos objetos, poucos atributos e uma baixa densidade: todos os paradigmas apresentam bons resultados, já que nesse caso o número total de conceitos formais é relativamente pequeno;
- poucos objetos, muitos atributos e uma densidade média: algoritmo *Godin*;
- muitos objetos, poucos atributos e uma densidade média: algoritmo *Next Neighbours*.

Note que tais sugestões são fundamentadas no arcabouço *EF - Concept Analysis*. Além disso, para cada paradigma apenas o principal representante foi analisado.

Outro ponto importante, que o leitor mais atento ainda pode questionar é a dimensão (número de registros e atributos) das bases de dados sintéticas e reais utilizadas para análise dos paradigmas, já que em problemas do mundo real a alta dimensionalidade é frequentemente observada. Entretanto, o objetivo desse trabalho é analisar os principais paradigmas de construção dos reticulados conceituais, apresentar uma forma bem definida para análise da gama de algoritmos existentes na literatura e propor novas abordagens para redução da complexidade existente no processo de obtenção do reticulado conceitual. Assim, um dos trabalhos futuros é a aplicação dos resultados apresentados nesse trabalho em aplicações com grande dimensionalidade.

Em relação à análise dos métodos de redução da complexidade, com os dois métodos propostos foi possível reduzir a complexidade do processo de construção do reticulado conceitual. Por exemplo, para a base de dados Water, a qual não foi possível gerar os conceitos formais com o contexto original, através do método RBA e JBS foi possível construir o reticulado conceitual selecionando apenas a informação considerada mais relevante. Os resultados demonstraram que as propostas apresentadas nesse trabalho são capazes de reduzir a complexidade do reticulado conceitual (número de conceitos formais). Além disso, é possível controlar a redução para atender os níveis desejáveis de qualidade do conhecimento ainda representado no reticulado conceitual.

Como supracitado, um ponto fraco desse trabalho é a falta de uma aplicação real na qual as propostas aqui abordadas possam ser aplicadas. Assim um primeiro trabalho futuro é a aplicação dos dois métodos de redução da complexidade propostos, juntamente com a metodologia proposta para avaliação desses métodos, em bases de dados reais de alta dimensionalidade. Além disso, é objetivo comparar em maiores detalhes as duas propostas de redução da complexidade do reticulado conceitual com outras abordagens para redução observadas na literatura.

Outra proposta é expandir o conceito de similaridade entre objetos para similaridade entre conceitos formais e aplicar o método de junção baseada em similaridade diretamente no reticulado conceitual; selecionando assim, apenas os conceitos formais mais similares. Além disso, pretende-se analisar a viabilidade da utilização de algoritmos de clusterização [Kantardzic, 2003] para determinar conjuntos de objetos similares.

Por último propõe-se melhorar as características do arcabouço *EF - Concept Analysis* e disponibilizá-lo a comunidade acadêmica para que o mesmo possa ser utilizado e avaliado. Juntamente com o arcabouço, pretende-se disponibilizar o *benchmark* pro-

posto nesse trabalho para que o mesmo possa ser incrementado e utilizado como referência para avaliação dos algoritmos e paradigmas de construção dos reticulados conceituais.

Anexo A

Informações Complementares das Bases de Dados Reais

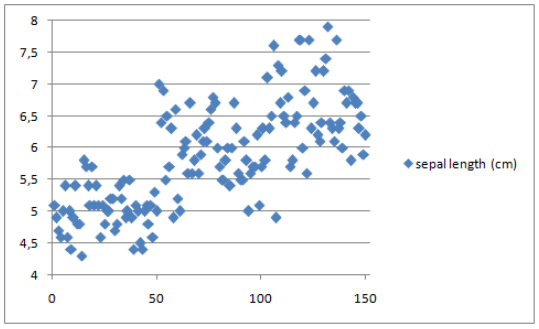

Figura A.1: Distribuição do atributo "sepal length (cm)" para a base de dados Iris.

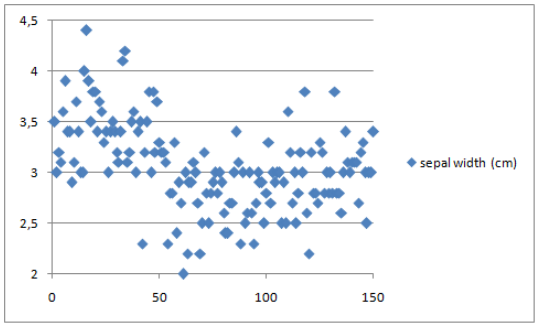

Figura A.2: Distribuição do atributo "sepal width (cm)" para a base de dados Iris.

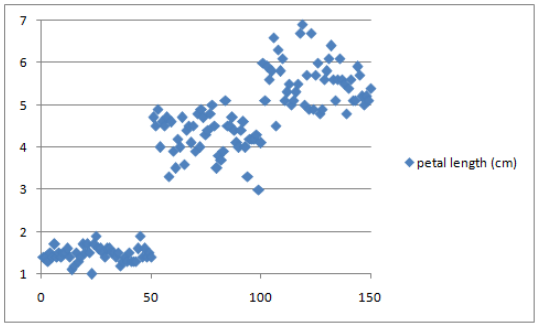

Figura A.3: Distribuição do atributo "petal length (cm)" para a base de dados Iris.

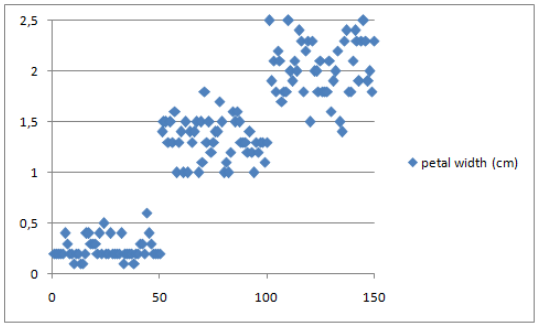

Figura A.4: Distribuição do atributo "petal width (cm)" para a base de dados Iris.

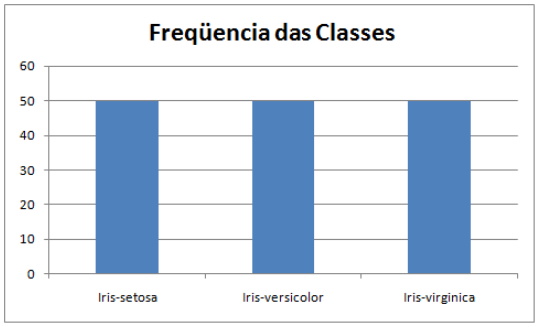

Figura A.5: Distribuição das classes para a base de dados Iris.

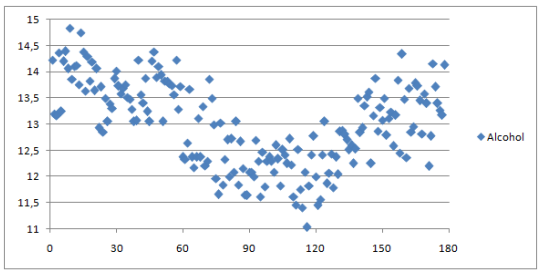

Figura A.6: Distribuição do atributo "Alcohol" para a base de dados Wine.

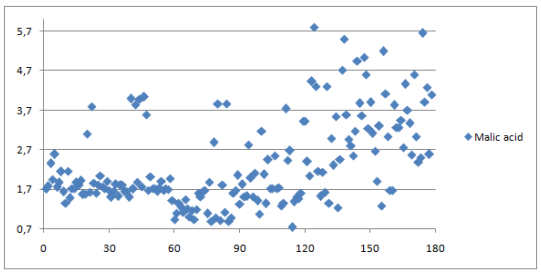

Figura A.7: Distribuição do atributo "Malic acid" para a base de dados Wine.

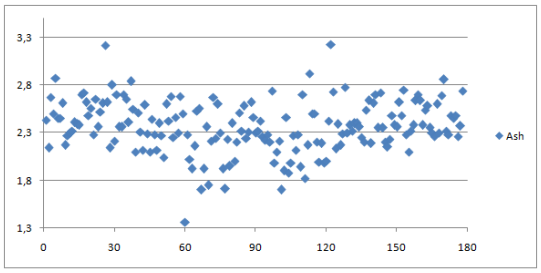

Figura A.8: Distribuição do atributo "Ash" para a base de dados Wine.

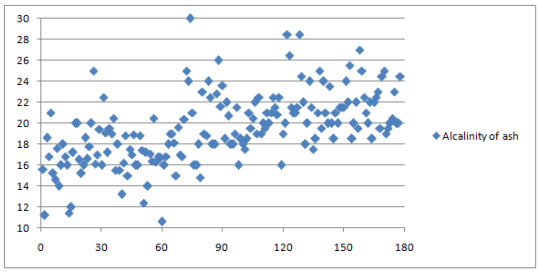

Figura A.9: Distribuição do atributo "Alcalinity of ash" para a base de dados Wine.

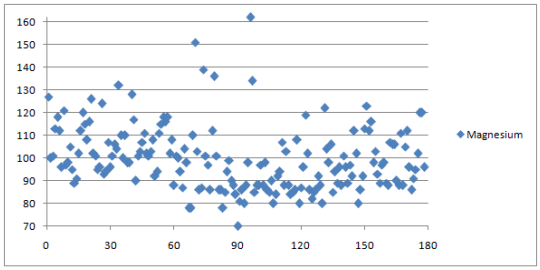

Figura A.10: Distribuição do atributo "Magnesium" para a base de dados Wine.

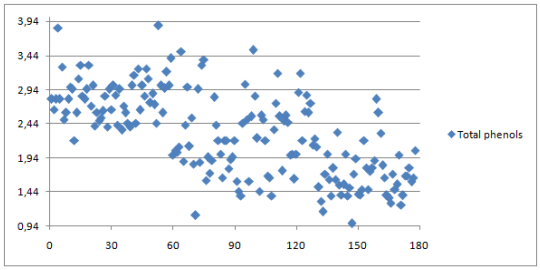

Figura A.11: Distribuição do atributo "Total phenols" para a base de dados Wine.

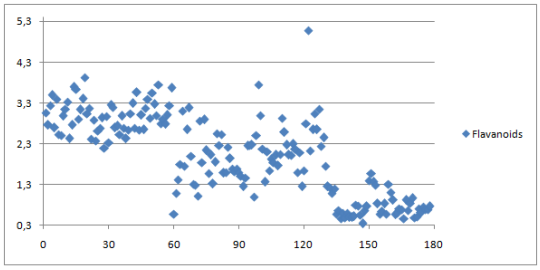

Figura A.12: Distribuição do atributo "Flavanoids" para a base de dados Wine.

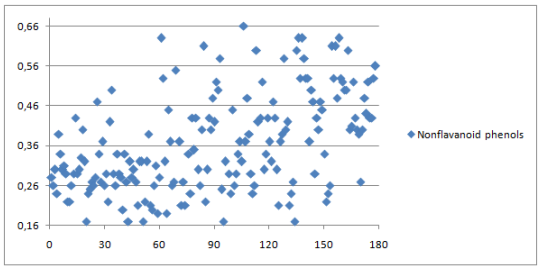

Figura A.13: Distribuição do atributo "Nonflavanoid phenols" para a base de dados Wine.

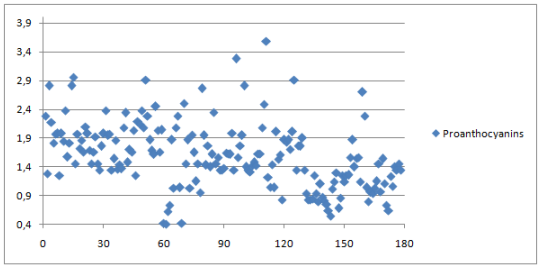

Figura A.14: Distribuição do atributo "Proanthocyanins" para a base de dados Wine.

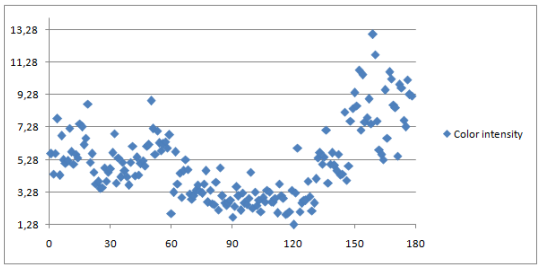

Figura A.15: Distribuição do atributo "Color intensity" para a base de dados Wine.

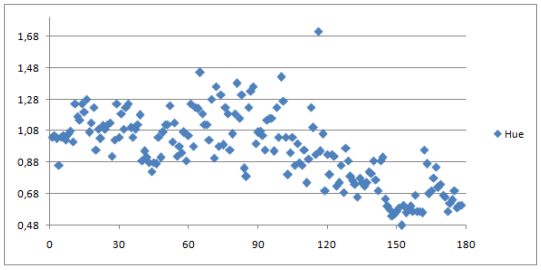

Figura A.16: Distribuição do atributo "Hue" para a base de dados Wine.

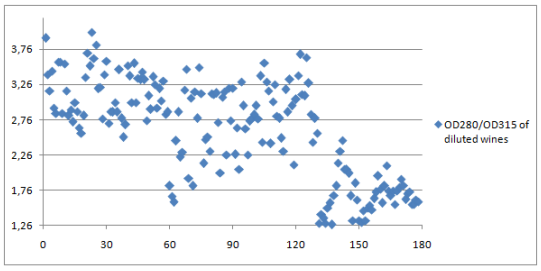

Figura A.17: Distribuição do atributo "OD280/OD315 of diluted wines" para a base de dados Wine.

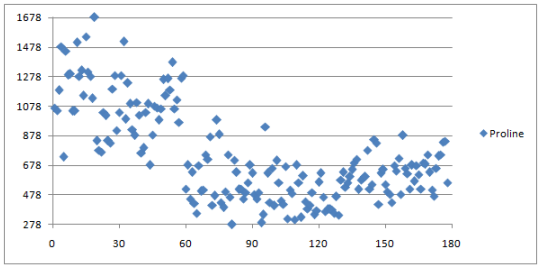

Figura A.18: Distribuição do atributo "Proline" para a base de dados Wine.

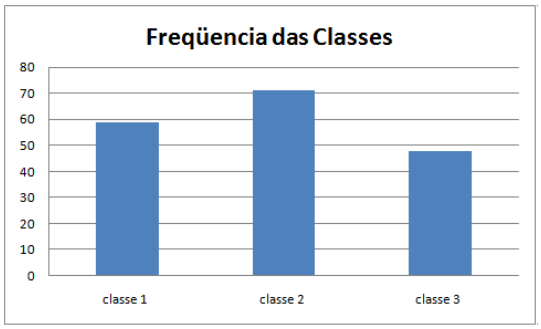

Figura A.19: Distribuição das classes para a base de dados Wine.

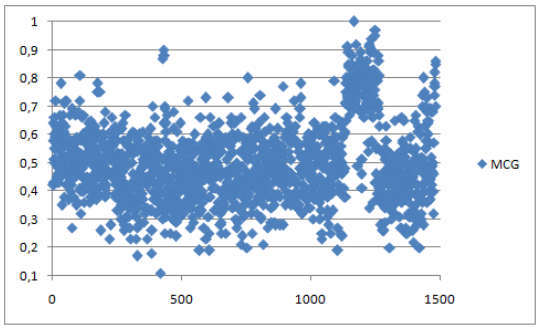

Figura A.20: Distribuição do atributo "MCG" para a base de dados Yeast.

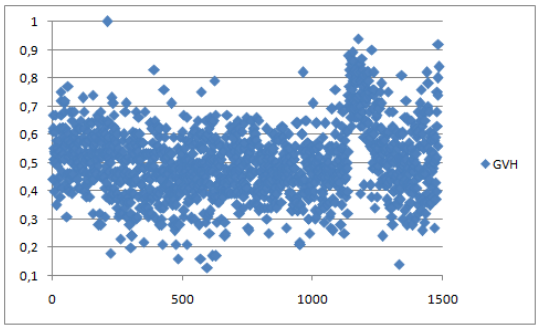

Figura A.21: Distribuição do atributo "GVH" para a base de dados Yeast.

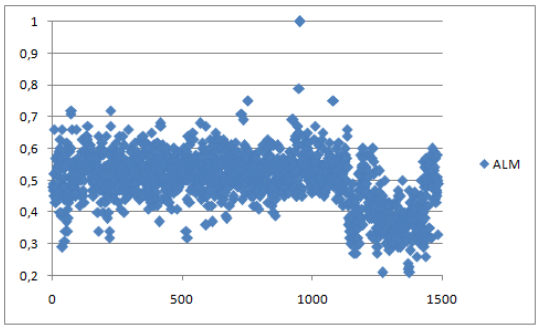

Figura A.22: Distribuição do atributo "ALM" para a base de dados Yeast.

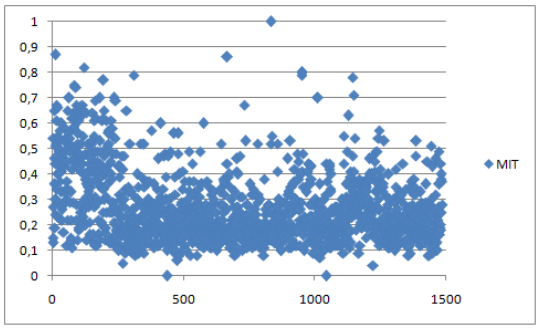

Figura A.23: Distribuição do atributo "MIT" para a base de dados Yeast.

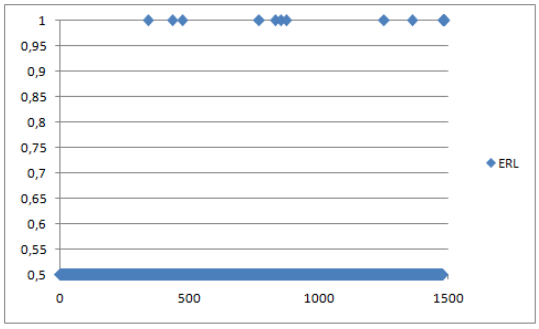

Figura A.24: Distribuição do atributo "ERL" para a base de dados Yeast.

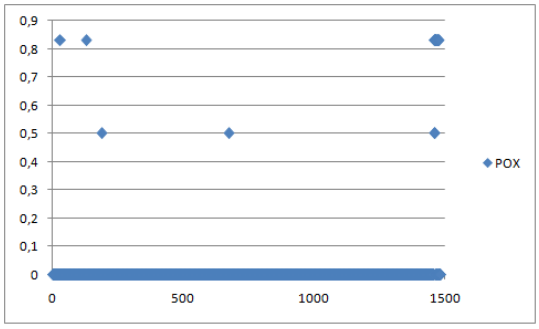

Figura A.25: Distribuição do atributo "POX" para a base de dados Yeast.

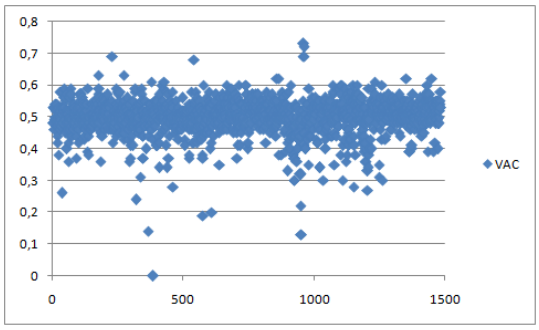

Figura A.26: Distribuição do atributo "VAC" para a base de dados Yeast.

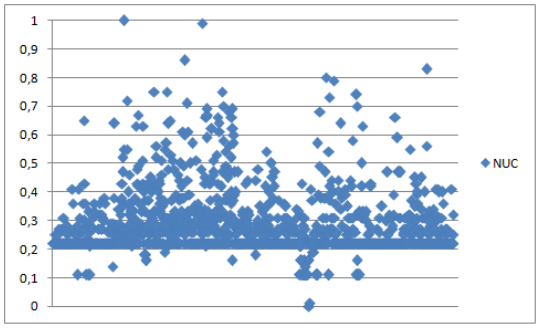

Figura A.27: Distribuição do atributo "NUC" para a base de dados Yeast.

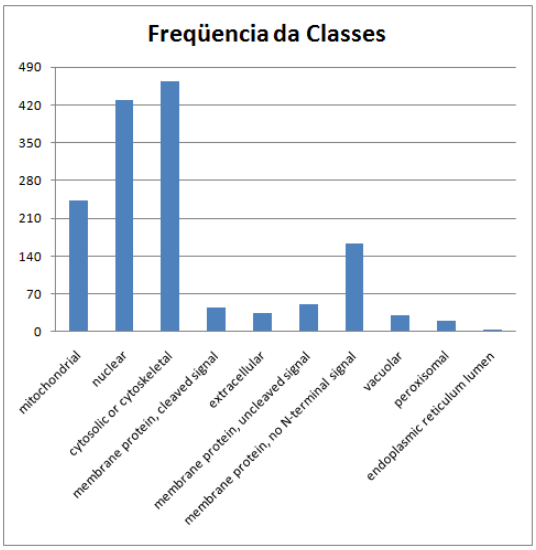

Figura A.28: Distribuição das classes para a base de dados Yeast.

Anexo B

Informações Complementares da Avaliação do Método de Junção Baseada em Similaridade

Tabela B.1: Base Íris aplicada no método JBS. Parte A

					Fidelidade				
|G|	**|I|**	**# Conceitos**	**# Regras**	ε	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
(%)	**(%)**	**Formais (%)**	**(%)**	**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
18	**40.27**	**104**	**67**	**0**	**100**	**100**	**100**	**0**	**0**
100.0	100.0	100.0	100.0	1	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	2	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	3	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	4	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	5	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	6	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	7	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	8	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	9	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	10	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	11	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	12	100.0	100.0	100.0	0.0	0.0
100.0	100.0	100.0	100.0	13	100.0	100.0	100.0	0.0	0.0
94.4	99.8	94.2	92.5	14	94.44	100	99.73	1.2	2.77
94.4	99.8	94.2	92.5	15	94.44	100	99.73	1.2	2.77
94.4	99.8	94.2	92.5	16	94.44	100	99.73	1.2	2.77
94.4	99.8	94.2	92.5	17	94.44	100	99.73	1.2	2.77
66.7	89.7	48.1	56.7	18	72.22	100	95.02	8.24	13.88
66.7	87.9	45.2	62.7	19	77.77	100	94.7	6.37	14.99
66.7	87.9	45.2	62.7	20	77.77	100	94.7	6.37	14.99
61.1	80.9	37.5	47.8	21	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	22	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	23	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	24	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	25	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	26	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	27	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	28	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	29	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	30	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	31	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	32	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	33	72.22	100	94.09	6.45	19.72
61.1	80.9	37.5	47.8	34	72.22	100	94.09	6.45	19.72
55.6	80.7	34.6	43.3	35	72.22	100	93.1	7.94	23.88
50.0	73.6	23.1	35.8	36	77.77	100	89.35	6.73	30.55
50.0	73.6	23.1	35.8	37	77.77	100	89.35	6.73	30.55
44.4	67.2	20.2	20.9	38	66.66	100	88.88	12.51	37.22
44.4	67.2	20.2	20.9	39	66.66	100	88.88	12.51	37.22
38.9	56.1	13.5	16.4	40	66.66	100	88.38	11.5	44.99
38.9	56.1	13.5	16.4	41	66.66	100	88.38	11.5	44.99
38.9	59.1	12.5	17.9	42	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	43	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	44	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	45	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	46	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	47	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	48	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	49	66.66	100	81.01	13.28	50.55
38.9	59.1	12.5	17.9	50	66.66	100	81.01	13.28	50.55

Tabela B.2: Base Íris aplicada no método JBS. Parte B

					Fidelidade				
|G| (%)	**|I| (%)**	**# Conceitos Formais (%)**	**# Regras (%)**	**ε (%)**	**Mín. (%)**	**Máx. (%)**	**Méd. (%)**	**Desv. Pad. (%)**	**Perda Desc. (%)**
18	**40.27**	**104**	**67**	**0**	**100**	**100**	**100**	**0**	**0**
33.3	51.7	9.6	13.4	51	66.66	100	81.48	13.6	58.05
33.3	51.7	9.6	13.4	52	66.66	100	81.48	13.6	58.05
33.3	51.7	9.6	13.4	53	66.66	100	81.48	13.6	58.05
33.3	51.7	9.6	13.4	54	66.66	100	81.48	13.6	58.05
33.3	51.7	9.6	13.4	55	66.66	100	81.48	13.6	58.05
38.9	59.1	12.5	20.9	56	66.66	94.44	82.14	10.49	52.49
27.8	33.1	5.8	6.0	57	72.22	94.44	84.72	11.45	66.94
27.8	33.1	5.8	6.0	58	72.22	94.44	84.72	11.45	66.94
27.8	33.1	5.8	6.0	59	72.22	94.44	84.72	11.45	66.94
27.8	33.1	6.7	6.0	60	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	61	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	62	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	63	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	64	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	65	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	66	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	67	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	68	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	69	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	70	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	71	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	72	72.22	88.88	83.33	7.85	66.94
27.8	33.1	6.7	6.0	73	72.22	88.88	83.33	7.85	66.94
33.3	41.4	6.7	10.4	74	50	88.88	74.6	13.92	65.83
33.3	41.4	6.7	10.4	75	50	88.88	74.6	13.92	65.83
33.3	34.5	5.8	7.5	76	50	88.88	70	13.94	75.55
33.3	34.5	5.8	7.5	77	50	88.88	70	13.94	75.55
27.8	24.8	3.8	4.5	78	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	79	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	80	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	81	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	82	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	83	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	84	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	85	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	86	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	87	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	88	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	89	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	90	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	91	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	92	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	93	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	94	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	95	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	96	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	97	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	98	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	99	72.22	94.44	81.48	11.56	80
27.8	24.8	3.8	4.5	100	72.22	94.44	81.48	11.56	80

Tabela B.3: Base Wine aplicada no método JBS. Parte A

					Fidelidade				
|G|	**|I|**	**# Conceitos**	**# Regras**	ε	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
(%)	**(%)**	**Formais (%)**	**(%)**	**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
132	**48.27**	**29031**	**40645**	**0**	**100**	**100**	**100**	**0**	**0**
100.00	100.00	100.00	100.00	1	100.0	100.0	100.0	0.0	0.0
100.00	100.00	100.00	100.00	2	100.0	100.0	100.0	0.0	0.0
100.00	100.00	100.00	100.00	3	100.0	100.0	100.0	0.0	0.0
100.00	100.00	100.00	100.00	4	100.0	100.0	100.0	0.0	0.0
100.00	100.00	100.00	100.00	5	100.0	100.0	100.0	0.0	0.0
100.00	100.00	100.00	100.00	6	100.0	100.0	100.0	0.0	0.0
100.00	100.00	100.00	100.00	7	100.0	100.0	100.0	0.0	0.0
77.27	97.20	70.75	79.64	8	95.5	100.0	99.6	0.6	3.8
77.27	97.20	70.75	79.64	9	95.5	100.0	99.6	0.6	3.8
77.27	97.20	70.75	79.64	10	95.5	100.0	99.6	0.6	3.8
77.27	97.20	70.75	79.64	11	95.5	100.0	99.6	0.6	3.8
77.27	97.20	70.75	79.64	12	95.5	100.0	99.6	0.6	3.8
75.00	96.25	63.05	72.02	13	94.7	100.0	99.5	0.7	4.9
75.00	96.25	63.05	72.02	14	94.7	100.0	99.5	0.7	4.9
50.76	85.60	18.46	28.33	15	87.1	100.0	97.5	2.2	18.8
50.76	85.60	18.46	28.33	16	87.1	100.0	97.5	2.2	18.8
50.76	85.60	18.46	28.33	17	87.1	100.0	97.5	2.2	18.8
50.76	85.60	18.46	28.33	18	87.1	100.0	97.5	2.2	18.8
50.76	85.60	18.46	28.33	19	87.1	100.0	97.5	2.2	18.8
50.00	83.65	15.80	24.14	20	86.4	100.0	97.2	2.4	20.1
50.00	83.65	15.80	24.14	21	86.4	100.0	97.2	2.4	20.1
34.09	70.15	3.73	6.42	22	78.0	100.0	93.7	4.1	34.3
34.09	70.15	3.73	6.42	23	78.0	100.0	93.7	4.1	34.3
34.09	70.15	3.73	6.42	24	78.0	100.0	93.7	4.1	34.3
34.09	70.15	3.73	6.42	25	78.0	100.0	93.7	4.1	34.3
34.09	70.15	3.73	6.42	26	78.0	100.0	93.7	4.1	34.3
34.85	68.63	3.30	5.82	27	73.5	100.0	93.5	4.3	35.8
34.85	67.85	3.15	5.46	28	73.5	100.0	93.3	4.3	36.4
24.24	52.89	0.83	1.35	29	68.9	100.0	89.5	7.0	49.9
24.24	52.89	0.83	1.35	30	68.9	100.0	89.5	7.0	49.9
24.24	52.89	0.83	1.35	31	68.9	100.0	89.5	7.0	49.9
24.24	52.89	0.83	1.35	32	68.9	100.0	89.5	7.0	49.9
24.24	52.89	0.83	1.35	33	68.9	100.0	89.5	7.0	49.9
23.48	51.38	0.76	1.20	34	68.9	100.0	89.3	7.2	51.8
23.48	49.76	0.71	1.14	35	68.9	100.0	88.9	7.1	53.2
16.67	36.69	0.23	0.33	36	63.6	100.0	83.9	9.2	66.0
16.67	36.69	0.23	0.33	37	63.6	100.0	83.9	9.2	66.0
16.67	36.69	0.23	0.33	38	63.6	100.0	83.9	9.2	66.0
16.67	36.69	0.23	0.33	39	63.6	100.0	83.9	9.2	66.0
16.67	36.69	0.23	0.33	40	63.6	100.0	83.9	9.2	66.0
16.67	35.05	0.19	0.31	41	63.6	100.0	82.5	8.6	67.5
16.67	34.74	0.19	0.30	42	63.6	100.0	82.1	8.3	68.1
12.88	24.78	0.08	0.07	43	59.8	100.0	84.7	11.0	74.6
12.88	24.78	0.08	0.07	44	59.8	100.0	84.7	11.0	74.6
12.88	24.78	0.08	0.07	45	59.8	100.0	84.7	11.0	74.6
12.88	24.78	0.08	0.07	46	59.8	100.0	84.7	11.0	74.6
12.88	24.78	0.08	0.07	47	59.8	100.0	84.7	11.0	74.6
12.88	23.10	0.08	0.06	48	59.8	100.0	84.0	11.3	76.6
12.12	20.97	0.07	0.05	49	59.8	100.0	84.0	11.0	79.5
8.33	18.83	0.04	0.02	50	64.4	100.0	80.0	12.4	79.9

Tabela B.4: Base Wine aplicada no método JBS. Parte B

					Fidelidade				
|G| (%)	**|I| (%)**	**# Conceitos Formais (%)**	**# Regras (%)**	**ε (%)**	**Mín. (%)**	**Máx. (%)**	**Méd. (%)**	**Desv. Pad. (%)**	**Perda Desc. (%)**
132	**48.27**	**29031**	**40645**	**0**	**100**	**100**	**100**	**0**	**0**
8.33	18.83	0.04	0.02	51	64.4	100.0	80.0	12.4	79.9
8.33	18.83	0.04	0.02	52	64.4	100.0	80.0	12.4	79.9
8.33	18.83	0.04	0.02	53	64.4	100.0	80.0	12.4	79.9
8.33	18.83	0.04	0.02	54	64.4	100.0	80.0	12.4	79.9
8.33	18.83	0.04	0.02	55	64.4	100.0	80.0	12.4	79.9
9.09	16.06	0.04	0.02	56	65.9	99.2	83.1	10.4	81.8
9.09	16.06	0.04	0.02	57	65.9	99.2	83.1	10.4	81.8
7.58	17.86	0.03	0.03	58	64.4	100.0	80.4	11.3	82.4
7.58	17.86	0.03	0.03	59	64.4	100.0	80.4	11.3	82.4
7.58	17.86	0.03	0.03	60	64.4	100.0	80.4	11.3	82.4
7.58	17.86	0.03	0.03	61	64.4	100.0	80.4	11.3	82.4
7.58	17.86	0.03	0.03	62	64.4	100.0	80.4	11.3	82.4
6.82	11.10	0.02	0.02	63	65.9	99.2	80.2	12.9	85.8
6.82	11.10	0.02	0.02	64	65.9	99.2	80.2	12.9	85.8
5.30	20.41	0.02	0.03	65	47.0	100.0	69.0	17.7	86.7
5.30	20.41	0.02	0.03	66	47.0	100.0	69.0	17.7	86.7
5.30	20.41	0.02	0.03	67	47.0	100.0	69.0	17.7	86.7
5.30	20.41	0.02	0.03	68	47.0	100.0	69.0	17.7	86.7
5.30	20.41	0.02	0.03	69	47.0	100.0	69.0	17.7	86.7
4.55	10.71	0.02	0.02	70	69.7	100.0	83.1	12.5	88.6
4.55	10.71	0.02	0.02	71	69.7	100.0	83.1	12.5	88.6
5.30	20.41	0.02	0.03	72	47.0	100.0	69.2	16.6	86.3
5.30	20.41	0.02	0.03	73	47.0	100.0	69.2	16.6	86.3
5.30	20.41	0.02	0.03	74	47.0	100.0	69.2	16.6	86.3
5.30	20.41	0.02	0.03	75	47.0	100.0	69.2	16.6	86.3
5.30	20.41	0.02	0.03	76	47.0	100.0	69.2	16.6	86.3
5.30	20.41	0.02	0.03	77	47.0	100.0	69.2	16.6	86.3
5.30	20.41	0.02	0.03	78	47.0	100.0	69.2	16.6	86.3
4.55	11.89	0.02	0.02	79	69.7	100.0	83.0	11.9	88.3
4.55	11.89	0.02	0.02	80	69.7	100.0	83.0	11.9	88.3
4.55	11.89	0.02	0.02	81	69.7	100.0	83.0	11.9	88.3
4.55	11.89	0.02	0.02	82	69.7	100.0	83.0	11.9	88.3
4.55	11.89	0.02	0.02	83	69.7	100.0	83.0	11.9	88.3
4.55	11.89	0.02	0.02	84	69.7	100.0	83.0	11.9	88.3
4.55	23.80	0.02	0.03	85	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	86	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	87	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	88	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	89	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	90	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	91	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	92	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	93	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	94	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	95	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	96	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	97	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	98	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	99	47.0	100.0	69.0	17.7	86.1
4.55	23.80	0.02	0.03	100	46.96	100	68.99	17.73	86.14

Tabela B.5: Base Yeast aplicada no método JBS. Parte A

					Fidelidade				
|G|	**|I|**	**# Conceitos**	**# Regras**	ε	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
(%)	**(%)**	**Formais (%)**	**(%)**	**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
110	**33.33**	**1929**	**1199**	**0**	**100**	**100**	**100**	**0**	**0**
98.2	99.9	99.3	98.9	1	99.09	100.00	99.99	0.07	0.40
84.5	99.2	94.6	91.3	2	94.54	100.00	99.94	0.40	2.42
75.5	98.2	85.1	77.3	3	93.63	100.00	99.75	0.87	4.14
69.1	95.9	64.6	70.1	4	91.81	100.00	99.38	1.29	6.46
53.6	94.2	52.7	54.4	5	84.54	100.00	99.02	1.96	8.58
38.2	92.1	43.3	33.0	6	83.63	100.00	98.66	3.42	10.20
38.2	92.1	43.3	33.0	7	83.63	100.00	98.66	3.42	10.20
38.2	92.1	43.3	33.0	8	83.63	100.00	98.66	3.42	10.20
38.2	92.1	43.3	33.0	9	83.63	100.00	98.66	3.42	10.20
38.2	92.1	43.3	33.0	10	83.63	100.00	98.66	3.42	10.20
40.0	92.5	44.0	36.4	11	84.54	100.00	98.26	3.52	9.69
45.5	90.3	39.2	42.2	12	89.09	100.00	98.29	1.96	10.60
46.4	88.7	40.0	39.7	13	88.18	100.00	98.11	2.02	12.92
44.5	86.9	37.8	38.4	14	86.36	100.00	98.29	2.27	15.45
42.7	81.6	27.3	16.6	15	83.63	100.00	96.62	3.65	20.00
38.2	80.8	26.6	15.7	16	84.54	100.00	96.87	3.72	20.80
30.9	78.5	24.0	11.8	17	78.18	100.00	96.79	4.44	22.22
30.9	78.5	24.0	11.8	18	78.18	100.00	96.79	4.44	22.22
30.9	78.5	24.0	11.8	19	78.18	100.00	96.79	4.44	22.22
30.9	78.5	24.0	11.8	20	78.18	100.00	96.79	4.44	22.22
30.9	78.5	24.0	11.8	21	78.18	100.00	96.79	4.44	22.22
31.8	77.8	23.7	11.7	22	78.18	100.00	96.35	4.70	23.43
31.8	64.2	13.5	15.4	23	71.81	100.00	93.61	5.83	45.35
27.3	56.7	8.2	10.7	24	69.09	100.00	91.43	6.57	52.62
25.5	54.4	6.8	8.3	25	58.18	100.00	90.65	8.39	55.25
25.5	46.8	4.3	5.7	26	59.09	98.18	87.68	9.61	61.41
21.8	48.6	4.9	9.0	27	56.36	100.00	87.92	10.45	63.33
17.3	42.7	2.7	5.2	28	53.63	99.09	85.26	10.59	66.06
17.3	42.7	2.7	5.2	29	53.63	99.09	85.26	10.59	66.06
17.3	42.7	2.7	5.2	30	53.63	99.09	85.26	10.59	66.06
17.3	42.7	2.7	5.2	31	53.63	99.09	85.26	10.59	66.06
17.3	42.7	2.7	5.2	32	53.63	99.09	85.26	10.59	66.06
17.3	42.7	2.7	5.2	33	53.63	99.09	85.26	10.59	66.06
18.2	40.6	2.4	2.9	34	50.90	99.09	82.90	13.10	63.63
20.0	40.9	2.4	3.0	35	57.27	97.27	85.07	11.62	61.91
20.0	37.9	2.5	2.3	36	58.18	97.27	86.36	10.93	61.51
17.3	27.5	1.2	1.3	37	42.72	93.63	74.37	21.07	76.96
17.3	26.3	1.4	1.5	38	40.90	96.36	76.96	20.63	75.55
12.7	21.4	0.7	1.2	39	51.81	92.72	79.02	12.18	84.64
12.7	21.4	0.7	1.2	40	51.81	92.72	79.02	12.18	84.64
12.7	21.4	0.7	1.2	41	51.81	92.72	79.02	12.18	84.64
12.7	21.4	0.7	1.2	42	51.81	92.72	79.02	12.18	84.64
12.7	21.4	0.7	1.2	43	51.81	92.72	79.02	12.18	84.64
12.7	21.4	0.7	1.2	44	51.81	92.72	79.02	12.18	84.64
12.7	17.5	0.8	0.9	45	35.45	94.54	78.18	18.84	85.45
12.7	17.5	0.6	0.8	46	35.45	94.54	78.63	20.11	84.54
10.0	21.2	0.7	0.8	47	33.63	91.81	72.82	18.30	83.83
10.0	7.1	0.3	0.2	48	91.81	92.72	92.27	0.64	91.21
10.9	8.3	0.3	0.2	49	91.81	92.72	92.27	0.64	91.61
8.2	6.2	0.2	0.1	50	91.81	91.81	91.81	0.00	92.92

Tabela B.6: Base Yeast aplicada no método JBS. Parte B

					Fidelidade				
|G| (%)	**|I| (%)**	**# Conceitos Formais (%)**	**# Regras (%)**	**ε (%)**	**Mín. (%)**	**Máx. (%)**	**Méd. (%)**	**Desv. Pad. (%)**	**Perda Desc. (%)**
110	**33.33**	**1929**	**1199**	**0**	**100**	**100**	**100**	**0**	**0**
8.2	6.2	0.2	0.1	51	91.81	91.81	91.81	0.00	92.92
8.2	6.2	0.2	0.1	52	91.81	91.81	91.81	0.00	92.92
8.2	6.2	0.2	0.1	53	91.81	91.81	91.81	0.00	92.92
8.2	6.2	0.2	0.1	54	91.81	91.81	91.81	0.00	92.92
8.2	7.4	0.3	0.1	55	91.81	91.81	91.81	0.00	91.91
8.2	8.6	0.3	0.2	56	91.81	92.72	92.27	0.64	90.90
8.2	8.6	0.3	0.2	57	91.81	92.72	92.27	0.64	90.90
9.1	8.9	0.3	0.2	58	91.81	92.72	92.27	0.64	91.51
10.0	7.1	0.3	0.2	59	91.81	92.72	92.27	0.64	92.62
8.2	6.2	0.2	0.2	60	91.81	92.72	92.27	0.64	93.83
5.5	7.4	0.2	0.1	61	91.81	91.81	91.81	0.00	92.32
5.5	7.4	0.2	0.1	62	91.81	91.81	91.81	0.00	92.32
5.5	7.4	0.2	0.1	63	91.81	91.81	91.81	0.00	92.32
5.5	7.4	0.2	0.1	64	91.81	91.81	91.81	0.00	92.32
5.5	7.4	0.2	0.1	65	91.81	91.81	91.81	0.00	92.32
5.5	7.4	0.2	0.1	66	91.81	91.81	91.81	0.00	92.32
6.4	6.3	0.2	0.1	67	91.81	91.81	91.81	0.00	93.63
6.4	6.3	0.2	0.1	68	91.81	91.81	91.81	0.00	93.63
6.4	6.3	0.2	0.1	69	91.81	91.81	91.81	0.00	93.63
5.5	5.6	0.2	0.1	70	91.81	91.81	91.81	0.00	94.54
5.5	5.6	0.2	0.1	71	91.81	91.81	91.81	0.00	94.54
4.5	4.4	0.1	0.2	72	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	73	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	74	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	75	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	76	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	77	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	78	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	79	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	80	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	81	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	82	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	83	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	84	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	85	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	86	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	87	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	88	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	89	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	90	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	91	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	92	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	93	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	94	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	95	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	96	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	97	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	98	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	99	83.63	91.81	87.72	5.78	95.55
4.5	4.4	0.1	0.2	100	83.63	91.81	87.72	5.78	95.55

Tabela B.7: Base Water aplicada no método JBS. Parte A

					Fidelidade				
|G|	**|I|**	**# Conceitos**	**# Regras**	ε	**Mín.**	**Máx.**	**Méd.**	**Desv.**	**Perda**
(%)	**(%)**	**Formais (%)**		**(%)**	**(%)**	**(%)**	**(%)**	**Pad. (%)**	**Desc. (%)**
380	**52.18**	**8940648**		**0**	**100**	**100**	**100**	**0**	**0**
31.33	78.59	2.14527	662247	14	64.24	100	95.98	4.77	21.88
25.32	71.92	0.90006	298141	15	50.31	100	93.37	7.35	28.67
25.32	71.92	0.90006	298141	16	50.31	100	93.37	7.35	28.67
25.32	71.92	0.90006	298141	17	50.31	100	93.37	7.35	28.67
18.35	63.53	0.22765	74007	18	37.34	100	90.9	11.28	37.95
18.35	63.53	0.22765	74007	19	37.34	100	90.9	11.28	37.95
15.82	57.09	0.12823	32945	20	35.12	100	91.56	12.74	44.05
15.82	57.09	0.12823	32945	21	35.12	100	91.56	12.74	44.05
15.82	57.09	0.12823	32945	22	35.12	100	91.56	12.74	44.05
12.03	49.12	0.04227	8921	23	26.58	100	91.88	13.74	51.35
12.03	49.12	0.04227	8921	24	26.58	100	91.88	13.74	51.35
12.03	49.12	0.04227	8921	25	26.58	100	91.88	13.74	51.35
9.18	41.53	0.01056	3139	26	31.96	100	89.33	14.31	58.27
9.18	41.53	0.01056	3139	27	31.96	100	89.33	14.31	58.27
9.18	41.53	0.01056	3139	28	31.96	100	89.33	14.31	58.27
8.54	34.57	0.00582	983	29	52.53	100	97.07	5.91	64.06
8.54	34.57	0.00582	983	30	52.53	100	97.07	5.91	64.06
8.54	34.57	0.00582	983	31	52.53	100	97.07	5.91	64.06
7.28	31.77	0.00407	714	32	45.56	100	97.19	6.54	67
7.28	31.77	0.00407	714	33	45.56	100	97.19	6.54	67
7.28	31.77	0.00407	714	34	45.56	100	97.19	6.54	67
5.38	28.02	0.00198	318	35	77.84	100	98.01	3.45	71.61
5.38	28.02	0.00198	318	36	77.84	100	98.01	3.45	71.61
5.38	28.02	0.00198	318	37	77.84	100	98.01	3.45	71.61
4.75	25.55	0.00158	290	38	79.43	100	96.64	4.55	73.67
4.75	25.55	0.00158	290	39	79.43	100	96.64	4.55	73.67
3.48	23.38	0.00086	205	40	83.22	100	95.23	5.39	75.12
3.48	23.38	0.00086	205	41	83.22	100	95.23	5.39	75.12
3.48	23.38	0.00086	205	42	83.22	100	95.23	5.39	75.12
2.85	21.58	0.00047	104	43	82.27	100	96.55	4.38	76.9
2.85	21.58	0.00047	104	44	82.27	100	96.55	4.38	76.9
2.85	21.58	0.00047	104	45	82.27	100	96.55	4.38	76.9
2.85	19.45	0.00038	65	46	86.39	100	97.17	3.54	79.21
2.85	19.45	0.00038	65	47	86.39	100	97.17	3.54	79.21
2.85	19.45	0.00038	65	48	86.39	100	97.17	3.54	79.21
2.53	18.47	0.00027	51	49	84.81	100	97.36	3.83	79.86
2.53	18.47	0.00027	51	50	84.81	100	97.36	3.83	79.86

Tabela B.8: Base Water aplicada no método JBS. Parte B

					Fidelidade				
|G| (%)	**|I| (%)**	**# Conceitos Formais (%)**	**# Regras**	**ε (%)**	**Mín. (%)**	**Máx. (%)**	**Méd. (%)**	**Desv. Pad. (%)**	**Perda Desc. (%)**
380	**52.18**	**8940648**		**0**	**100**	**100**	**100**	**0**	**0**
2.53	18.47	0.00027	51	51	84.81	100	97.36	3.83	79.86
2.53	18.47	0.00027	47	52	85.44	100	96.32	4.55	80.72
2.53	18.47	0.00027	47	53	85.44	100	96.32	4.55	80.72
2.53	18.47	0.00027	47	54	85.44	100	96.32	4.55	80.72
1.90	16.86	0.00023	42	55	85.12	100	96.1	5.43	82.57
1.90	16.86	0.00023	42	56	85.12	100	96.1	5.43	82.57
1.90	16.86	0.00023	42	57	85.12	100	96.1	5.43	82.57
1.90	30.11	0.00016	93	58	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	59	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	60	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	61	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	62	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	63	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	64	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	65	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	66	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	67	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	68	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	69	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	70	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	71	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	72	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	73	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	74	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	75	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	76	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	77	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	78	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	79	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	80	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	81	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	82	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	83	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	84	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	85	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	86	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	87	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	88	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	89	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	90	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	91	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	92	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	93	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	94	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	95	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	96	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	97	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	98	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	99	2.21	100	40.47	39.83	82.02
1.90	30.11	0.00016	93	100	2.21	100	40.47	39.83	82.02

Referências Bibliográficas

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H. e Verkamo, A. I. (1996). Fast discovery of association rules. *American Association for Artificial Intelligence*, pp. 307–328.

Arévalo, G.; Berry, A.; Huchard, M.; Perrot, G. e Sigayret, A. (2007). Performances of galois sub-hierarchy-building algorithms. In *International Conference on Formal Concept Analysis - ICFCA*, pp. 166–180.

Asuncion, A. e Newman, D. (2009). UCI machine learning repository. <http://www.ics.uci.edu/∼mlearn/MLRepository.html> último acesso Janeiro de 2009.

Axler, S. e Ribet, K. A. (2005). *Lattices and Ordered Algebraic Structures*. Springer-Verlag, London.

Belohlávek, R.; Sklenar, V. e Zacpal, J. (2004). Concept lattices constrained by attribute dependencies. In *DATESO*, volume 98 of *CEUR Workshop Proceedings*, pp. 63–73. CEUR-WS.org.

Ben-Khalifa, K. e Motameny, S. (2007). Horn-representation of a concept lattice. In Eklund, P. W.; Diatta, J. e Liquiere, M., editores, *Fifth International Conference on Concept Lattices and Their Applications*, volume 331 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Bertet, K.; Guillas, S. e Ogier, J.-M. (2007). Extensions of Bordat's algorithm for attributes. In *Fifth International Conference on Concept Lattices and Their Applications*, pp. 34–45.

Bordat, J. P. (1986). Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et sciences humaines*, 24(96):31–47.

Brunner, J. (1990). *Acts of Mining*. Cambridge, MA: Harvard University Press.

Buchli, F. (2003). Detecting software patterns using formal concept analysis. Technical report, Diplomarbeit der Philosophisch-naturwissenschaftlichen Fakultat der Universitat Bern. http://www.iam.unibe.ch/publikationen/techreports/2003/iam-03-010.

Carpineto, C. e Romano, G. (1993). Galois: An order-theoretic approach to conceptual clustering. In *ICML*, pp. 33–40.

Carpineto, C. e Romano, G. (2004a). *Concept Data Analysis: Theory and Applications*. John Wiley & Sons, England.

Carpineto, C. e Romano, G. (2004b). Exploiting the potential of concept lattices for information retrieval with credo. *J. UCS*, 10(8):985–1013.

Carpineto, C.; Romano, G. e D'adamo, P. (1999). Inferring dependencies from relations: A conceptual clustering approach. *Computational Intelligence*, 15:15–4.

Chein, M. (1969). Algorithme de recherche des sous-matrices premières d'une matrice. *Bull. Math. Soc. Sc. Math. de Roumanie*, 1(13):21–25.

Cheung, K. S. K. e Vogel, D. (2005). Complexity reduction in lattice-based information retrieval. *Information Retrieval*, 8(2):285–299.

Davey, B. e Priestley, H. (1990). *Introduction to lattices and order.* Cambridge University Press, Cambridge, England.

Fay, G. (1973). An algorithm for finite Galois connections. Technical report, Institute for Industrial Economy, Budapest.

Fu, H. e Nguifo, E. M. (2003). Partitioning large data to scale up lattice-based algorithm. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 537–544.

Fu, H. e Nguifo, E. M. (2004). A lattice algorithm for data mining. Technical report, Université d' Artois-IUT de Lens.

Gajdos, P.; Moravec, P. e Snásel, V. (2004). Concept lattice generation by singular value decomposition. In *CLA*, pp. 102 –110.

Ganter, B. (1984). Two basic algorithms in concept analysis. Technical report, TU Darmstadt, Germany.

Ganter, B. e Stumme, G. (2003). Formal concept analysis: Methods and applications in computer science. Technical report, Otto - von - Guericke - Universitat Magdeburg. último acesso em Janeiro de 2009.

Ganter, B. e Wille, R. (1998). *Applied lattice theory: Formal concept analysis.* Springer-Verlag, Germany.

Ganter, B. e Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Germany.

Godin, R. e Chau, T. (2000). Comparaison d'algorithmes de construction de hiérarchies de classes. *L'objet*, 5.

Godin, R. e Missaoui, R. (1994). An incremental concept formation approach for learning from databases. *Theoretical Computer Science*, 133(2):387–419.

Godin, R.; Missaoui, R. e Alaoui, H. (1991). Learning algorithms using galois lattice structure. In *Proceedings of the IEEE International Conference on Tools for Artificial Intelligence*, pp. 22–29.

Godin, R.; Missaoui, R. e Alaoui, H. (1995). Incremental concept formation algorithms based on galois (concepts) lattices. *Computation Intelligence*, 11.

Godin, R.; Saunders, E. e Gecsei, J. (1986). Lattice model of browsable data spaces. *Inf. Sci.*, 40(2):89–116.

Guénoche, A. (1990). Construction du treillis de galois d'une relation binaire. *Mathématiques et Sciences Humaines*, (109):41–53.

Harzheim, E. (2005). *Ordered Sets.* Springer-Verlag, London.

Jay, N.; Kohler, F. e Napoli, A. (2008). Analysis of social communities with iceberg and stability-based concept lattices. In *International Conference on Formal Concept Analysis - ICFCA*, pp. 258–272.

Kantardzic, M. (2003). *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons.

Kovács, L. (2004). Concept lattice structure with attribute lattice. *Publication of University of Miskolc*, 10(8):985–1013.

Kuznetsov, S. (1990). Stability as an estimate of the degree of substantiation of hypotheses derived on the basis of operational similarity. *Nauchn. Tekh. Inf*, (12):21–29. in Russian.

Kuznetsov, S. (2001). On computing the size of a lattice and related decision problems. *Order*, 18:313–321(9).

Kuznetsov, S. e Obiedkov, S. (2001). Algorithms for the construction of concept lattices and their diagram graphs. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, volume 2168, pp. 289–300. Lecture Notes In Computer Science.

Kuznetsov, S. e Obiedkov, S. (2002). Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14:189–216.

Kuznetsov, S.; Obiedkov, S. e Roth, C. (2007). Reducing the representation complexity of lattice-based taxonomies. In Priss, U.; Polovina, S. e Hill, R., editores, *Proceedings of the 15th International Conference on Conceptual Structures (ICCS 2007)*, volume 4604 of *Lecture Notes in Artificial Intelligence*, pp. 241–254, Berlin, Heidelberg. Springer-Verlag.

Malgrange, Y. (1962). Recherche des sous-matrices premères d'une matrice à coeffientes binaires; application à cetains problémes de graphe. In *Deuxième congrès de L'AFCALTI*, pp. 231–242.

Martin, B. e Eklund, P. W. (2008). From concepts to concept lattice: A border algorithm for making covers explicit. In *International Conference on Formal Concept Analysis - ICFCA*, pp. 78–89.

Nation, J. B. (200?). Notes on lattice theory. Technical report, University of Hawaii, Hawaii.

Norris, E. M. (1978). An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250.

Nourine, L. e Raynaud, O. (1999). A fast algorithm for building lattices. *MInformation Processing Letters*, 71:199 – 204.

Old, J. e Priss, U. (2006). Some open problems in formal concept analysis. Problems presented at International Conference on Formal Concept Analysis (ICFCA) 2006 in Dresden.

Priss, U. (2009). Formal concept analysis software. <http://www.upriss.org.uk/fca/fcasoftware.html> último acesso Janeiro de 2009.

Qiao, S.-Y.; Wen, S.-P.; Chen, C.-Y. e Li, Z.-G. (2003). A fast algorithm for building concept lattice. In *Machine Learning and Cybernetics International Conference*, volume 1 Issue , 2-5, pp. 163 – 167.

Roth, C.; Obiedkov, S. e Kourie, D. G. (2006). Towards concise representation for taxonomies of epistemic communities. In *CLA 4th International Conference on Concept Lattices and their Applications, Tunis, Faculté des Sciences de Tunis*, pp. 205–218.

Schutt, D. (1998). Abschatzungen fur die anzahl der begridde von kontexten. Master's thesis, TH Darmstandt, Gemany.

Stumme, G. (2002). Formal concept analysis on its way from mathematics to computer science. In *ICCS*, Lecture Notes in Computer Science, pp. 2–19, London, UK. Springer-Verlag.

Stumme, G.; Taouil, R.; Bastide, Y.; Pasquier, N. e Lakhal, L. (2002). Computing iceberg concept lattices with titanic. *Data and Knowledge Engineering*, 42:189–222(34).

Traub, J. F. e Wozniakowski, H. (1994). Breaking intractability. *j-SCI-AMER*, 270(1):102–?? (Intl. ed. 90–93).

Tsopze, N.; Nguifo, E. M. e Tindo, G. (2007). Clann: Concept lattice-based artificial neural networks for supervised classification. In Diatta Jean, E. P. e Michel, L.,

editores, *Intl. Conf. on Concept Lattices and their Applications(CLA07)*, pp. 157–168, Montpellier (France).

Tsopze, N.; Nguifo, E. M. e Tindo, G. (2008). M-clann: Multi-class concept lattice-based artificial neural network for supervised classification. In et al., V. K., editor, *18th International Conference on Artificial Neural Networks(ICANN-08)*, Prague (RTC). Springer-Verlag LNAI. September 3-6; to appear.

Valtchev, P. e Missaoui, R. (2002). A partition-based approach towards constructing galois (concept) lattices. *Discrete Mathematics*, 256:801–829.

Vimieiro, R. (2007). Um estudo de algoritmos para a extração de regras baseados em análise formal de conceitos. Master's thesis, Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação, Belo Horizonte, Brazil.

Wang, X. e Zhang, W. (2008). Relations of attribute reduction between object and property oriented concept lattices. *Knowledge-Based Systems*, 21(5):398–403.

Wenxiu, Z.; Ling, W. e Jianjun, Q. (2005). Attribute reduction theory and approach to concept lattice. *Science in China Series F: Information Sciences*.

Wille, R. (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. *I. Rival (Ed.): Ordered Sets*, pp. 445–470.

Wille, R. (2005). Formal concept analysis as mathematical theory of concepts and concept hierarchies. In *Formal Concept Analysis*, pp. 1–33.

Wolskil, M. (2005). Formal concept analysis and rough set theory from the perspective of finite topological approximations. In *Transactions on Rough Sets III*, volume 3400, pp. 230–243. Springer Berlin / Heidelberg.

Xie Cui-hua, Li Yun, S. J. C. J.-j. e Jian-li, L. (2005). A constructing algorithm of concept lattice with attribute generalization based on cloud models. In *The Fifth International Conference on Computer and Information Technology*, pp. 110–115.

Yang, Y. e Webb, G. I. (2005). *Encyclopedia of data warehousing and mining*. J. Wang. Idea Group Reference.

Yevtushenko, S. A. (2000). System of data analysis “concept explorer”. In *Proceedings of the 7th national conference on Artificial Intelligence KII-2000*, pp. 127–134, Russia.

Zárate, L. E.; Dias, S. M. e Song, M. A. J. (2008). Fcann: A new approach for extraction and representation of knowledge from ann trained via formal concept analysis. *Neurocomputing*, 71:2670–2684.

Zhou, W.; tian Liu, Z. e Zhao, Y. (2007). Ontology learning by clustering based on fuzzy formal concept analysis. In *Computer Software and Applications Conference*, volume 1, pp. 204 – 210.