UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MINERAÇÃO DE PREFERÊNCIAS CONTEXTUAIS *FUZZY*

JULIETE APARECIDA RAMOS COSTA

Uberlândia - Minas Gerais
2014

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JULIETE APARECIDA RAMOS COSTA

MINERAÇÃO DE PREFERÊNCIAS CONTEXTUAIS *FUZZY*

Dissertação de Mestrado apresentada à Faculdade de Ciência da Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Banco de Dados.

Orientadora:
Prof[a]. Dr[a]. Sandra Aparecida de Amo

Uberlândia, Minas Gerais
2014

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Ciência da Computação a aceitação da dissertação intitulada "**Mineração de Preferências Contextuais *Fuzzy***" por **Juliete Aparecida Ramos Costa** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 23 de 05 de 2014

Orientadora: _____
Profª. Drª. Sandra Aparecida de Amo
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Profª. Drª. Marilde Terezinha Prado Santos
Universidade Federal de São Carlos

_____
Profª. Drª. Denise Guliato
Universidade Federal de Uberlândia

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: 23 de 05 de 2014

Autor:	**Juliete Aparecida Ramos Costa**
Título:	**Mineração de Preferências Contextuais *Fuzzy***
Faculdade:	**Faculdade de Ciência da Computação**
Grau:	**Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

Dedicatória

Aos meus pais, Raimunda e Jovenilto, e meus irmãos Éder, Helder e Heliton.
Aos meus sobrinhos Ruan e Isabella.

Agradecimentos

A Deus, por me fortalecer nos momentos mais difíceis percorridos nessa caminhada.

Aos meus pais Jovenilton e Raimunda, meus irmãos Éder, Helder e Heliton, meus sobrinhos Ruan e Isabella e as minhas cunhadas Simone e Kênia por terem me apoiado durante esta caminhada e por compreenderem a minha ausência em tantos momentos.

A minha orientadora *Profª Sandra*, pela oportunidade, orientação, paciência, ensinamentos e profissionalismo durante o desenvolvimento deste trabalho. Especialmente por ter acreditado em mim e pelo crescimento profissional que me proporcionou durante este período.

Aos meus tios Wagner e Lucinéia, pelo apoio no momento que precisei no início desta etapa. Agradecendo eles, agradeço a toda minha família pelo apoio, orações e incentivo durante esta caminhada.

A todos os meus verdadeiros amigos que mesmo longe sempre estiveram comigo. Em especial a Isabel e Laerte Allan, sem vocês o início deste trabalho não seria possível. Aos meus mestres Giuliano Viana e Felipe César por terem acreditado em mim. E um agradecimento especial a minha amiga Santana Coutinho que mesmo sem me conhecer me acolheu no início desta caminhada.

Aos meus colegas de laboratório, meus companheiros de luta diária que se tornaram verdadeiros amigos. Obrigada pelo incentivo e cooperação durante o desenvolvimento deste trabalho. Em especial ao Romerson, Taffarel, Rafael, Jaqueline, Joicy, Luciane, Dani, Crícia, Marcos, Guilherme, Myllene e Cris.

As minhas amigas Geycy, Cleiane e Emanuelle que compartilharam comigo vários momentos ímpares durante este período. Agradeço também às minhas amigas Marianna, Bruna, Natália e Cíntia pelo companheirismo e amizade sincera durante esta jornada.

À Família GPP-Uberlândia, pelas orações, intercessões e amizade durante o desenvolvimento deste trabalho. E um agradecimento especial ao secretário da Pós-Graduação Erisvaldo, pelo carinho, amizade e gentileza que sempre me atendeu.

A CAPES pelo apoio financeiro durante o desenvolvimento deste trabalho.

"Tu és, Senhor, o meu pastor, por isso nada em minha vida faltará."
(Salmo 23, 1)

Resumo

Nos últimos anos, muitas pesquisas na área de mineração de preferências são focadas no desenvolvimento de métodos para minerar um modelo de preferência a partir de representações de preferências *crisp*, tais como, conjunto de pares de *tuplas*, *ranking* de *tuplas* e funções *score*. Nesta pesquisa, é apresentado o algoritmo FuzzyPrefMiner, desenvolvido para minerar um modelo de preferência contextual *fuzzy* a partir de uma representação de preferência denominada *relação de preferência fuzzy*. Este tipo de representação é composta por um conjunto de *triplas* do tipo (u, v, n), onde u e v são *tuplas* avaliadas pelo usuário e n é o grau de preferência da *tupla* u sobre a *tupla* v.

As relações de preferências *fuzzy* possibilitam ainda, o estudo da consistência das preferências dos usuários e neste contexto, são apresentados dois métodos de reparação de inconsistência dessas relações, nomeados: *Range Voting* Não Incremental e *Range Voting* Incremental. Ambos os métodos são baseados em uma técnica de sistema de votação e são utilizados para reparar a inconsistência das relações de preferências *fuzzy* inferidas pelo FuzzyPrefMiner. Uma série de experimentos em dados reais foram realizados, para validar a proposta do algoritmo FuzzyPrefMiner e dos métodos de reparação de inconsistência. Os resultados obtidos se mostram bastante satisfatórios, quando comparados com o estado da arte em relação a algoritmos de mineração de preferências e métodos de reparação de inconsistência.

Palavras chave: preferências contextuais, relações de preferências *fuzzy*, mineração de preferências, métodos de reparação de inconsistência.

Abstract

In recent years, much researches in Preference Mining has focused on development of methods for mining a preference model from crisp pairwise representation. In this research, is presented the *FuzzyPrefMiner* algorithm, developed for mining fuzzy contextual preference model from fuzzy preference relation. This type of preference representation is composed by a set of triples (u, v, n), where, u and v are tuples evaluated by user and n is the degree of preference of tuple u with respect to v.

The fuzzy preference relations enable the study of the consistency of users and in this context, are presented two methods for repair inconsistency of these relations: *No Incremental Range Voting* and *Incremental Range Voting*. Both methods are based on technical Voting System and are used for repair the inconsistency of fuzzy preference relations inferred by *FuzzyPrefMiner*. A set of experiments in real data were performed to validate the *FuzzyPrefMiner* algorithm and the methods for repair inconsistency. The results quite satisfactory when compared with the state of the art regarding Preference Mining Algorithms and Methods for Repair Inconsistency.

Keywords: contextual preferences, fuzzy preference relations, preference mining, methods for repair inconsistency.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

acc	Acurácia
AG	Algoritmo Genético
BDP-C	Banco de Dados de Preferências *Crisp*
BDP-F	Banco de Dados de Preferências *Fuzzy*
BTF	Baseado em Transitividade Fraca
C	Ciclos
CP-Net	*Contextual Preference Network*
CPrefMiner	*Contextual Preference Mining*
d	Dia
d'	Diferença
$\bar{d}'$	Diferença Média
FuzzyPrefMiner	*Fuzzy Preference Mining*
gp	Grau de Preferência
gp_{max}	Grau de Preferência Máximo
gp_{min}	Grau de Preferência Mínimo
gp_p	Grau de Preferência Predito
gp_r	Grau de Preferência Real
HA	Hipótese Alternativa
HO	Hipótese Nula
hrs	Horas
mc	Média de Ciclos
mc_e	Média de Ciclos de uma Matriz Esparsa
mc_c	Média de Ciclos de uma Matriz Cheia
mseg	Milissegundos
PNet	*Preference Network*
$PREF$	Preferência
$PREF_{rv}$	Preferência do *Range Voting*

prec	Precisão
RPB	Rede de Preferência *Bayesiana*
RPB-c	Rede de Preferência *Bayesiana* do cenário *Crisp*
RPB-f	Rede de Preferência *Bayesiana* do cenário *Fuzzy*
RV	*Range Voting*
revoc	Revocação
seg	Segundos
ta	Taxa de Aleatoriedade
tc	Taxa de Comparabilidade
p	Porcentagem de Preenchimento da Matriz

Capítulo 1

Introdução

O grande aumento de dados e a velocidade com que esse volume cresce pode causar frustrações aos usuários, quando o mesmo necessita obter alguma informação de maneira rápida e seletiva. Neste contexto, nos últimos 10 anos o tópico de Mineração de Preferências tem atraído bastante atenção na comunidade de Pesquisa em Mineração de Dados, devido a seu importante papel no desenvolvimento de Sistemas de Recomendação (Burges *et al.*, 2005), (Lops *et al.*, 2011).

Um algoritmo de mineração de preferências trabalha na extração de um modelo de preferências *acurado* a partir de um conjunto de amostras de preferências do usuário. Os formalismos mais comuns usados para representar essas amostras de preferências são funções *scores* (Burges *et al.*, 2005), (Lops *et al.*, 2011), (Crammer e Singer, 2001), pares de tuplas (de Amo *et al.*, 2012b), (de Amo *et al.*, 2012a), (Koriche e Zanuttini, 2010) e *ranking* de tuplas (Freund *et al.*, 2003), (Joachims, 2002).

A maioria das técnicas de mineração de dados utilizadas em sistemas de recomendação são as técnicas de classificação tradicionais. Neste cenário, o conjunto de treinamento é constituído por tuplas avaliadas pelo usuário, onde as notas são consideradas *classes* e o classificador é usado para inferir a nota que um usuário daria a uma nova tupla. No entanto, dependendo da aplicação o uso do classificador para predizer notas do usuário não é viável. Por exemplo, quando um usuário avalia um filme, a classificação dada normalmente não está relacionada apenas às características do filme, mas também às avaliações feitas pelo usuário em outros filmes.

Portanto, a informação importante neste contexto não é a classificação exata dos filmes, mas como o usuário compara dois filmes. Assim, para algumas aplicações, as técnicas de mineração de preferências usadas para extrair um modelo de preferências a partir de um conjunto de pares de tuplas são mais adequadas.

O trabalho proposto em de Amo *et al.* (2012a) implementa um algoritmo nomeado CPrefMiner para minerar um modelo de preferências contextual *crisp*, a partir de um conjunto de amostras de preferências do usuário representado por pares de tuplas. O modelo de preferências minerado pelo algoritmo é considerado *crisp*, pois ele é extraído de um conjunto de dados onde o usuário fornece sua preferência através de uma resposta *"sim"* ou *"não"*. Isto é, ou o usuário prefere a tupla u a tupla v (denotado por $u \succ v$), ou ele prefere a tupla v a tupla u (denotado por $v \succ u$). Logo, o modelo de preferências extraído pelo CPrefMiner prediz se uma nova tupla w_1 é preferível a uma nova tupla w_2 ou se w_2 é preferível a tupla w_1.

O modelo de preferências extraído pelo CprefMiner é *contextual*, ou seja, ele é constituído por um conjunto de *regras de preferências contextuais* (Wilson, 2004) da forma: *"SE <contexto> ENTÃO EU prefiro "valor 1" a "valor 2"*. Por exemplo, uma regra de preferência contextual no cenário de filmes poderia ser composta da seguinte forma: *Para os filmes cujo ator é Tom Hanks (contexto), eu prefiro "Comédia" a "Ação"*. Dessa forma, o modelo de preferências extraído pelo CprefMiner é capaz de inferir uma *relação de ordem* sobre as tuplas (o que a tarefa de classificação não é capaz de alcançar), isto é, o modelo é capaz de inferir relacionamentos de *transitividade* (se $u \succ v$ e $v \succ w$ então $u \succ w$) satisfazendo a propriedade de *irreflexibilidade* (if $u \succ v$ então $v \not\succ u$).

Técnicas de mineração de preferências que usam como dados de entrada um conjunto de pares de tuplas, normalmente executam uma etapa de pré-processamento para transformar avaliações feitas pelo usuário em pares de tuplas. Por exemplo, se o usuário avalia uma tupla u com nota **5** e a tupla v com nota **1**, esta informação é transformada no par (u, v). Neste tipo de transformação, uma grande quantidade de informação é perdida, pois, se o usuário avaliasse as tuplas u e v com notas **5** e **4** respectivamente, o mesmo par (u, v) seria formado. No entanto, na primeira situação o usuário prefere a tupla u a tupla v com uma intensidade maior do que na segunda opção.

Com base nessas informações, esta dissertação apresenta a proposta do algoritmo FuzzyPrefMiner, para minerar um modelo de preferências contextual *fuzzy* a partir de um conjunto de amostras de preferências representadas por *triplas* (u, v, n), onde u e v são tuplas avaliadas pelo usuário e n é o *grau de preferência* $(0 \leq n \leq 1)$ da tupla u sobre a tupla v. Logo, o modelo de preferências extraído é considerado *fuzzy*, pois consegue inferir o *grau de preferência* sobre duas novas tuplas w_1 e w_2.

Para realizar o pré-processamento dos dados e obter um conjunto de *triplas* do tipo (u, v, n) é necessário transformar as avaliações das tuplas em uma relação de preferência *fuzzy* (Chiclana *et al.*, 1998). Essa relação é representada por uma matriz quadrada e cada posição da matriz indica o *grau de preferência* de uma tupla sobre a outra e este é obtido por funções de transformações matemáticas (Tanino, 1984), (Chiclana *et al.*, 1998). A partir da matriz de preferências é possível avaliar a consistência do usuário a partir das propriedades de transitividade das relações de preferências *fuzzy*, tais como, transitividade

fraca e transitividade aditiva (Herrera-Viedma *et al.*, 2004), (Herrera-Viedma *et al.*, 2007).

As funções matemáticas utilizadas nesta dissertação garantem a consistência das relações de preferências *fuzzy* pré-processadas para criar o conjunto de *triplas* (u, v, n). No entanto, a relação de preferência *fuzzy* constituída dos *graus de preferência* inferidos pelo algoritmo FuzzyPrefMiner não atendem essas propriedades. Neste contexto, este trabalho de dissertação apresenta também, a proposta de um método de reparação de inconsistência das relações de preferências *fuzzy* inferidas pelo FuzzyPrefMiner. Tal método é baseado na técnica de sistema de votação *Range Voting* (de Amo *et al.*, 2014) e consiste em utilizar as predições feitas pelo FuzzyPrefMiner, para tentar eliminar todas as inconsistências de uma relação de preferência *fuzzy*. Diante disso, o objetivo geral e objetivos específicos deste trabalho de dissertação são:

- Objetivo Geral
  - Mostrar que o *grau de preferência* é um fator importante para mineração de um modelo de preferências mais acurado.
- Objetivos Específicos
  - Formalizar o problema de mineração de preferências *crisp* no cenário *fuzzy*.
  - Desenvolver o algoritmo FuzzyPrefMiner proposto para solucionar o problema de mineração *fuzzy*.
  - Analisar o desempenho do algoritmo FuzzyPrefMiner em comparação com um trabalho do estado da arte, em relação à mineração de modelos de preferências contextuais, através de medidas de avaliação como: acurácia, revocação, precisão, taxa de comparabilidade e taxa de aleatoriedade.
  - Desenvolver métodos de reparação de inconsistência de relações de preferências *fuzzy*.
  - Analisar o desempenho do método de reparação de inconsistência proposto, em comparação com o estado da arte, em relação a métodos de reparação de inconsistência de relações de preferências *fuzzy* através das medidas de avaliação.

Diante dos objetivos aqui citados, as principais contribuições deste trabalho são:

- Formalização do Problema de Mineração *Fuzzy* (Capítulo 5).
- Desenvolvimento do algoritmo FuzzyPrefMiner para mineração de modelos de preferências contextuais *fuzzy* (Capítulo 5).
- Estudo da Influência do Grau de Preferência em algoritmos de mineração de preferências contextuais (Capítulo 5 e Capítulo 7).
- Definição, implementação e validação de métodos de reparação de inconsistência em relações de preferências *fuzzy* (Capítulo 6 e Capítulo 7).

Esta dissertação encontra-se organizada como descrito abaixo e a Figura 1.1 apresenta a sequência de leitura deste trabalho e as contribuições do trabalho estão destacadas nos retângulos com cor escura.

- O Capítulo 2 introduz os conceitos preliminares necessários para compreender a proposta do algoritmo FuzzyPrefMiner, o contexto que o mesmo é aplicado e algumas noções de consistência em preferências *fuzzy*.
- No Capítulo 3 são apresentados os trabalhos relacionados ao trabalho proposto nesta dissertação.
- No Capítulo 4 são apresentados, a formalização do problema de mineração de preferências no cenário *crisp*, bem como, o algoritmo CPrefMiner (de Amo *et al.*, 2012a), pois, o mesmo é a base para desenvolvimento do FuzzyPrefMiner e o principal *baseline* de comparação.
- O Capítulo 5 apresenta a formalização do problema de mineração de preferências *fuzzy* e apresentação do algoritmo FuzzyPrefMiner.
- O Capítulo 6 apresenta os métodos de reparação de inconsistência de relações de preferências *fuzzy* propostos nesta dissertação e o método considerado como *baseline* de comparação.
- O Capítulo 7 apresenta os resultados experimentais obtidos sobre dados reais, tanto para o algoritmo FuzzyPrefMiner, quanto para os métodos de reparação de inconsistência;
- Finalmente, o Capítulo 8 apresenta as conclusões e trabalhos futuros para este trabalho de dissertação.

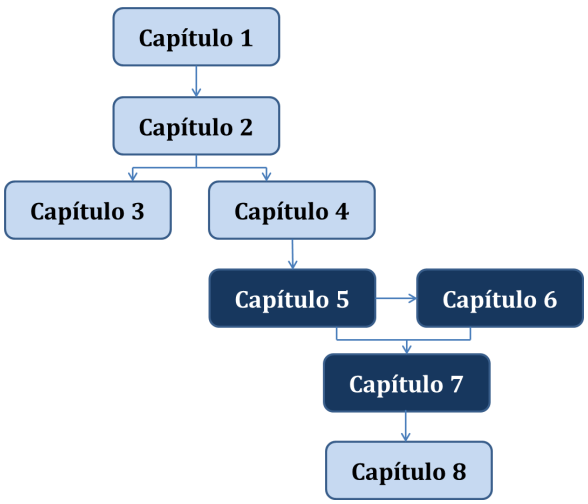


Figura 1.1: Diagrama de Leitura da Dissertação

Capítulo

2

Conceitos Preliminares

Este capítulo tem como objetivos: (1) descrever o problema geral de um algoritmo de Mineração de Preferências, (2) apresentar diversas formas de representação de preferências *crisp* e *fuzzy*, (3) mostrar algumas formas de transformar representações de preferências, (4) apresentar os aspectos dos modelos extraídos por algoritmos de mineração de preferências e (5) finalmente, apresentar algumas noções de consistência em preferências *fuzzy*.

Desta forma, a Seção 2.1 descreve os principais aspectos envolvidos no problema geral de Mineração de Preferências. Em seguida, a Seção 2.2 aborda alguns tipos comuns de representações *quantitativas* de preferências *crisp* do usuário e seus principais conceitos. A Seção 2.3 mostra representações de preferências *fuzzy* utilizadas neste trabalho de dissertação. A Seção 2.4 aborda funções de transformação de representações de preferências. A Seção 2.5 aborda alguns aspectos sobre tipos de modelos de preferências e a Seção 2.6 destaca algumas noções de consistência em relações de preferências *fuzzy*.

2.1 Mineração de Preferências

Antes de mostrar os diversos tipos de representação de preferências, é importante mostrar os principais aspectos envolvidos no problema geral de Mineração de Preferências no qual essa pesquisa se engloba.

No problema geral de Mineração de Preferências ilustrado pela Figura 2.1, um algoritmo de Mineração recebe como entrada algum tipo de representação *quantitativa* de preferências de um usuário "u", tais como: *ranking* de tuplas, pares de tuplas, conjunto de *triplas* e função *score*. Essas representações são consideradas *quantitativas*, pois envolve tudo aquilo que o usuário avaliou.

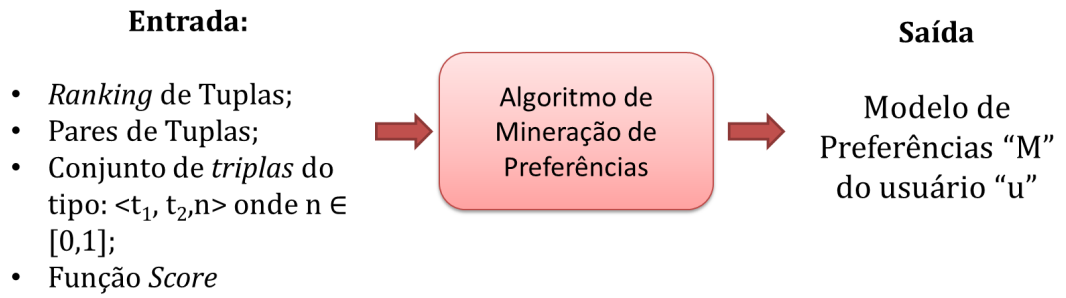


Figura 2.1: Diagrama Geral: Algoritmo de Mineração de Preferências - Representando a etapa de extração do modelo de preferências

O algoritmo retorna como saída algum tipo de representação de preferências *qualitativa*, definida pelo Modelo de Preferências "M"do usuário "u". Este modelo é considerado *qualitativo* por ser um modelo mais compacto e não possuir todas as avaliações do usuário, mas sim, regras capazes de predizer a preferência sobre novas tuplas.

Existem algoritmos que retornam como saída representações *quantitativas*, no entanto, o poder de expressividade da especificação *qualitativa* é maior que na especificação *quantitativa*, uma vez que, nem todas as relações de preferências podem ser expressas por funções *scores* ou *ranking* de tuplas. De um modo geral, a abordagem *qualitativa* proporciona regras mais compactas e fáceis de armazenar, podendo ser utilizadas para predizer a preferência sobre tuplas não avaliados pelo usuário.

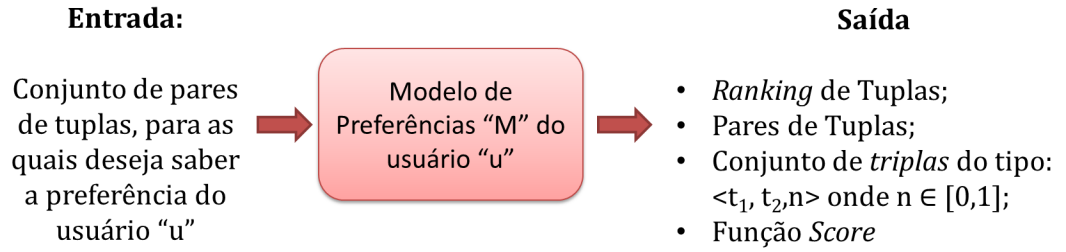


Figura 2.2: Diagrama Geral: Modelo de Preferências - Representando a etapa de uso do modelo de preferências extraído pelo algoritmo de mineração

O Modelo de Preferências "M"obtido pelo algoritmo de mineração de preferências, como visto na Figura 2.2, é capaz de predizer a preferência de um usuário "u"sobre duas tuplas que ele ainda não avaliou. O conceito de preferências aparece na entrada do algoritmo de mineração (Figura 2.1) e na saída do Modelo de Preferências "M"(Figura 2.2), e ambas são representações *quantitativas*. Daí a necessidade de estabelecer maneiras de representações de preferências do usuário. É disto exatamente que se trata a próxima seção deste capítulo.

2.2 Representações de Preferências *Crisp*

Preferências podem ser expressas de forma *qualitativa* (de Amo *et al.*, 2012a), (Holland *et al.*, 2003) ou *quantitativa* (Cohen *et al.*, 1999). Em Stefanidis *et al.* (2011), vários trabalhos envolvendo essas duas abordagens são apresentados. Na abordagem *qualitativa* um Modelo de Preferência é extraído por um algoritmo e este modelo é utilizado para predizer a preferência entre novas tuplas. Na abordagem *quantitativa* o algoritmo retorna uma representação quantitativa de preferência para todas as tuplas, por exemplo, uma nota para cada tupla do banco de dados. Esse processo se torna inviável quando se tem um grande conjunto de dados. Esta seção aborda algumas formas de representação de preferências *crisp* do usuário.

2.2.1 Pares de tuplas *Crisp*

Esse tipo de representação de preferências também conhecida como *bituplas* pode ser vista em de Amo *et al.* (2012a). Neste trabalho, um conjunto de *bituplas* forma um banco de dados de preferências *crisp*, definido da seguinte forma:

Definição 2.1. (Banco de Dados de Preferências *Crisp* (BDP-C)) Seja $R = (A_1, A_2, ..., A_n)$ um esquema relacional e $Tup(R)$ o conjunto de tuplas sobre R. Um banco de dados de preferência *crisp* sobre R é um conjunto finito $P \subseteq Tup(R) \times Tup(R)$ que é consistente, ou seja, se $(u, v) \in P$ então $(v, u) \notin P$. O par (u, v) representa o fato de que entre as tuplas u e v, o usuário prefere u.

Id	A	B	C	D
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_1	d_2
t_3	a_2	b_1	c_1	d_2
t_4	a_1	b_2	c_1	d_2
t_5	a_2	b_1	c_2	d_1
t_6	a_3	b_1	c_1	d_1

(a)

(t_1,t_2)
(t_1,t_3)
(t_4,t_5)
(t_4,t_2)
(t_5,t_6)
(t_3,t_5)
(t_4,t_1)

(b)

Figura 2.3: Banco de Dados de Preferências *Crisp*

A Figura 2.3b ilustra um exemplo de banco de dados de preferências sobre um esquema relacional $R = (A, B, C, D)$, com os domínios definidos por $\text{dom}(A) = \{a_1, a_2, a_3\}$, $\text{dom}(B) = \{b_1, b_2\}$, $\text{dom}(C) = \{c_1, c_2\}$ e $\text{dom}(D) = \{d_1, d_2\}$. O banco de dados de preferência é representado por uma instância de tuplas (Figura 2.3a). Neste banco de

dados de preferências, é possível observar que a tupla $t_1(a_1, a_1, c_1, d_1)$ é preferível a tupla $t_2(a_1, b_1, c_1, d_2)$.

2.2.2 *Ranking* de Tuplas

Essa forma de representação de preferências do usuário, descrita em Chiclana *et al.* (1998) e Chiclana *et al.* (2001), consiste em uma ordenação (*ranking*) de tuplas definida por:

Definição 2.2. (*Ranking* de Tuplas) Seja $R(A_1, A_2, ..., A_j)$ um esquema relacional e $Tup(R)$ uma lista de n tuplas $[t_1, \ldots, t_n]$ sobre R. Um *ranking* é uma permutação r de $[1, \ldots, n]$, tal que, a lista de tuplas $[t_1, \ldots, t_n]$ está ordenada por ordem de preferência do usuário.

Id	A	B	C	D
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_1	d_2
t_3	a_2	b_1	c_1	d_2
t_4	a_1	b_2	c_1	d_2
t_5	a_2	b_1	c_2	d_1
t_6	a_3	b_1	c_1	d_1

(a)

r
t_3
t_1
t_4
t_2
t_6
t_5

(b)

Figura 2.4: *Ranking* de Tuplas

Um exemplo de *ranking* de tuplas é ilustrado na Figura 2.4b. Neste *ranking*, pode-se observar que a tupla t_1 é preferível a tupla t_2, pois t_1 ocupa a segunda posição no *ranking*, enquanto t_2 ocupa a quarta posição. Através do *ranking* é possível notar também, que t_3 é a tupla mais preferida, pois ocupa a primeira posição do *ranking*, enquanto t_5 é a tupla menos preferida ocupando a última posição do *ranking*.

2.2.3 Função *Score*

Representar preferências do usuário por uma função *score* é simplesmente associar a cada tupla do banco de dados, um valor que representa a nota que usuário avalia tal tupla.

Definição 2.3. (Função *Score* (Chiclana *et al.*, 1998)) Seja $R(A_1, A_2, ..., A_j)$ um esquema relacional e $Tup(R)$ um conjunto de tuplas sobre R. Um usuário fornece suas preferências sobre $Tup(R)$ através de uma função $u : Tup(R) \rightarrow N$, que associa para cada tupla em $Tup(R)$ uma nota N

Id	A	B	C	D
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_1	d_2
t_3	a_2	b_1	c_1	d_2
t_4	a_1	b_2	c_1	d_2
t_5	a_2	b_1	c_2	d_1
t_6	a_3	b_1	c_1	d_1

(a)

Tup(R)	N
t_1	8
t_2	4
t_3	9
t_4	7
t_5	3
t_6	6

(b)

Figura 2.5: Função *Score* sobre $Tup(R)$

No exemplo de função *score* ilustrado pela Figura 2.5(b), é notável que o usuário prefere a tupla t_3 a todas as outras tuplas da instância, pois sua nota é maior que a nota das outras tuplas. Já t_5 é a tupla menos preferida, pois, tem menor valor de função *score*.

Uma vez destacadas formas de representações de preferências *crisp* do usuário, faz-se necessário, apresentar na próxima seção algumas definições de representações de preferências *fuzzy*.

2.3 Representações de Preferências *Fuzzy*

Nesta seção são apresentadas algumas formas de representação de preferências *fuzzy*. Essas representações são usadas neste trabalho de dissertação para representar as preferências do usuário.

2.3.1 Pares de tuplas *Fuzzy*

Representar preferências do usuário por pares de tuplas *fuzzy*, consiste em obter os pares de tuplas como definido na Seção 2.2.1, associando a cada par um valor *fuzzy* que representa o grau de preferência entre as tuplas. Sendo assim, defini-se Banco de Dados de Preferências *Fuzzy* como se segue:

Definição 2.4. (Banco de Dados de Preferências *Fuzzy*(BDP-F)) Um BDP-F sobre um esquema relacional $R = (A_1, ..., A_k)$ é um conjunto finito $\mathcal{P} \subseteq Tup(R) \times Tup(R) \times [0, 1]$ que é consistente, isto é, se (u, v, n)e$(v, u, m) \in \mathcal{P}$, então $m = 1 - n$. A *tripla* (u, v, n) representa o fato do usuário preferir a tupla u a tupla v com grau de preferência n.

A Figura 2.6(b) ilustra um banco de dados de preferência *fuzzy* sobre um conjunto de tuplas (Figura 2.6(a)). É possível perceber na figura que, o usuário prefere a tupla t_1 a tuplas $t2$ com grau de preferência 0.83. Neste tipo de representação a resposta do usuário não é apenas "*sim*"ou "*não*", ele identifica o *grau de preferência* entre as tuplas que estão sendo comparadas.

Id	A	B	C	D
t_1	a_1	b_1	c_1	d_1
t_2	a_1	b_1	c_1	d_2
t_3	a_2	b_1	c_1	d_2
t_4	a_1	b_2	c_1	d_2
t_5	a_2	b_1	c_2	d_1
t_6	a_3	b_1	c_1	d_1

(a)

(t_1,t_2,0.83)
(t_1,t_3,0.72)
(t_4,t_5,0.64)
(t_4,t_2,0.91)
(t_5,t_6,0.69)
(t_3,t_5,0.75)
(t_4,t_1,0.86)

(b)

Figura 2.6: Banco de Dados de Preferências *Fuzzy*

2.3.2 Relação de Preferência *Fuzzy*

Relação de Preferência *Fuzzy* é uma forma mais rica de representação de preferências. Neste tipo de representação, também é possível inferir *o quanto* uma tupla é preferível a outra, isto é, é possível saber o *grau de preferência* entre duas tuplas distintas. Basicamente uma relação de preferência *fuzzy* (Chiclana *et al.*, 1998) é definida da seguinte forma:

Definição 2.5. (Relação de Preferência *Fuzzy*) Seja $R(A_1, A_2, ..., A_n)$ um esquema relacional e $Tup(R)$ o conjunto de todas as *tuplas* sobre R avaliadas por um usuário u. Uma relação de preferência *fuzzy* sobre $Tup(R)$ é uma matriz $M_{n\times n}$, tal que, os valores $m_{ij} \in M$ são obtidos por uma função de transformação $f : Tup(R) \times Tup(R) \rightarrow [0.1]$. A função $f(t_i, t_j) = m_{ij}$ denota o *grau de preferência* (gp) da tupla t_i sobre a tupla t_j: $m_{ij} = 0.5$ indica indiferença entre as tuplas t_i e t_j e $m_{ij} > 0.5$ indica que t_i é preferível a t_j com o $gp = m_{ij}$.

Exemplo 2.1. Suponha o seguinte conjunto de tuplas $Tup(R) = \{t_1, t_2, t_3, t_4\}$. De alguma maneira o usuário informa suas preferências pela seguinte relação de preferência *fuzzy*:

$$M = \begin{bmatrix} \mathbf{0.5} & 0.61 & 0.86 & 0.96 \\ 0.39 & \mathbf{0.5} & 0.8 & 0.94 \\ 0.14 & 0.2 & \mathbf{0.5} & 0.8 \\ 0.04 & 0.06 & 0.2 & \mathbf{0.5} \end{bmatrix}$$

A relação de preferência *fuzzy* M possui o *grau de preferência* de cada *tupla* em $Tup(R)$ quando comparada com todas as demais *tuplas*, inclusive com ela mesmo. Neste exemplo, nota-se que foi gerada uma matriz 4×4, ou seja, todas as *tuplas* são comparados com todas as outras *tuplas* do conjunto $Tup(R)$.

Com este tipo de representação de preferências, é possível analisar a intensidade de preferência (positivas (>0.5) e negativas (<0.5)) entre as *tuplas* e consequetemente verificar as possíveis inconsistência do usuário.

Nota-se uma relação entre as definições de representações de preferências *fuzzy* destacadas nesta seção. Portanto, a partir de uma relação de preferência *fuzzy* é possível obter um banco de dados de preferência *fuzzy* e vice-versa.

2.4 Transformações de Representações de Preferências

Uma vez estabelecido os tipos de representações de preferências *crisp* e *fuzzy*, é necessário mostrar como transformar representações de preferências *crisp* em representações *fuzzy*.

Vários tipos de representação de preferências não-*fuzzy* podem ser transformadas em preferências *fuzzy*, por exemplo, *ranking* de tuplas e função *score* e podem ser vistas com mais detalhes em Chiclana *et al.* (1998) e Chiclana *et al.* (2001). O foco dessa pesquisa está voltado para a transformação de preferências por função *score* em relação de preferências *fuzzy*, portanto, esta seção aborda com mais detalhes esse tipo de transformação.

O objetivo aqui é mostrar como transformar notas que avaliam tuplas em uma relação de preferência *fuzzy*. Isso é possível através de funções matemáticas que atendam algumas propriedades importantes. Existem na literatura várias funções que transformam avaliações de tuplas em relações de preferências *fuzzy*, dentre elas, se destacam a família de funções baseadas na escala de razão entre as notas de duas tuplas. Tais funções foram propostas inicialmente por Tanino (1984) e depois por Chiclana *et al.* (1998).

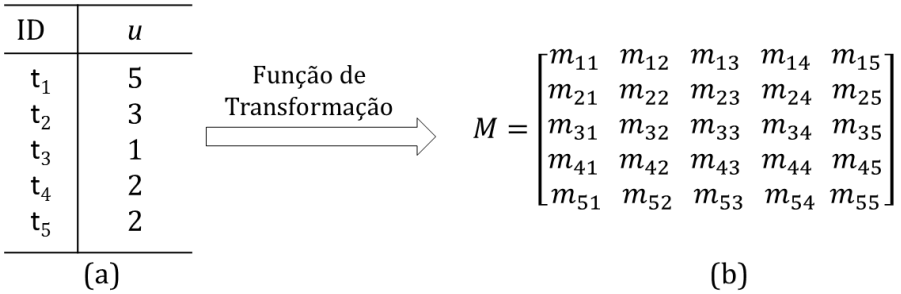


Figura 2.7: Esquema Geral - Transformação de Notas de tuplas em Matriz de Preferência

Discussão: Transformar Notas de tuplas em Relações de Preferência *Fuzzy*

Suponha o conjunto de tuplas $Tup = \{t_1, t_2, t_3, t_4, t_5\}$ avaliados por um usuário "y"através de alguma função *score* $u : Tup(R) \Rightarrow \{1, 2, 3, 4, 5\}$, onde, $\{1, 2, 3, 4, 5\} \in dom(u)$ como pode ser visto na Figura 2.7a.

É preciso encontrar uma função que transforme esse tipo de representação, provinda de uma função *score* em uma relação de preferência *fuzzy* representada por uma matriz $M_{5\times5}$, como visto na Figura 2.7b. Nesta figura, o valor m_{ij} representa o *grau de preferência* da tupla t_i sobre a tupla t_j. Além disso, a relação de preferência *fuzzy* M obtida por uma função de transformação deve atender as seguintes propriedades:

1. $m_{ij} \in [0,1]$: Essa propriedade é importante para estabelecer um limite, isto é, um intervalo fixo para todos os valores da matriz M.
2. $m_{ij} > 0.5 \Leftrightarrow u_i > u_j$: Essa propriedade indica que a tupla t_i é preferível a tupla t_j. Isso é indicado pelo valor $m_{ij} > 0.5$, que será encontrado somente se a nota da tupla t_i for maior que a nota de t_j.
3. $m_{ij} + m_{ji} = 1$: Essa propriedade indica a reciprocidade entre os *graus de preferências*. Significa que o valor m_{ji}, isto é, o grau de preferência de t_j sobre t_i é o valor complementar de m_{ij} no intervalo $[0,1]$.
4. $m_{ij} = 0.5 \Leftrightarrow u_i = u_j$: Essa propriedade indica indiferença entre duas tuplas. O valor 0.5 é encontrado sempre que se compara uma tupla com ela mesma, ou quando os duas tuplas possuem a mesma nota.

Tendo em mente as propriedades que os elementos que uma relação de preferência *fuzzy* devem atender, é preciso encontrar uma forma de obter os valores m_{ij} que compõem a matriz M, a partir de um conjunto de tuplas avaliados pelo usuário. As próximas subseções mostram como realizar essa transformação. A subseção 2.4.1 mostra uma maneira inicial para realizar essa transformação e a subseção 2.4.2 apresenta um refinamento dessa forma inicial que resulta em uma função de transformação que atende essas propriedades das relações de preferência *fuzzy*.

2.4.1 Obtenção dos Valores da Matriz de Preferência

Uma primeira forma de encontrar os valores m_{ij} de uma matriz de preferência M, consiste em utilizar uma função baseada na escala de razão $f_1 = \frac{u_i}{u_j}$ entre as notas de duas tuplas t_i e t_j. Essa proposta compara duas notas e retorna um único valor representando o *grau de preferência* de uma tupla sobre outra. Para compreender essa como funciona essa transformação, considere um exemplo:

Exemplo 2.2. Suponha o mesmo conjunto de *tuplas* do exemplo anterior, ou seja, $Tup = \{t_1, t_2, t_3, t_4, t_5\}$ avaliados pelo usuário por uma função *score* $u : Tup(R) \Rightarrow \{1,2,3,4,5\}$ (veja Figura 2.8a).

Para transformar essas notas em uma relação de preferência *fuzzy*, a função $f_1 = \frac{u_i}{u_j}$ é utilizada. Com essa função, os valores m_{ij} de uma matriz de preferência $M_{5\times5}$ são obtidos, como ilustra a Figura 2.8b.

Na Figura 2.8(b), é possível notar que a primeira propriedade de uma relação de preferência *fuzzy* não é satisfeita. Essa primeira propriedade diz que todo valor $m_{ij} \in [0,1]$ e os valores da matriz M, obtidos pela função $f_1 = \frac{u_i}{u_j}$ não estão neste intervalo. Por exemplo, a posição m_{13} da matriz tem valor 5. Logicamente, as demais propriedades também não são satisfeitas, pois todas elas envolvem valores no intervalo [0,1].

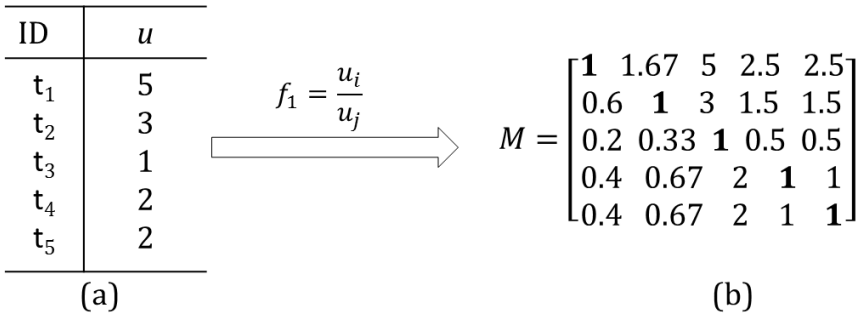


Figura 2.8: Esquema Geral - Obtendo o *grau de preferência* usando a função $f_1 = \frac{u_i}{u_j}$

É notável, que o resultado obtido pela função baseada somente na escala de razão não atende as propriedades de uma relação de preferência *fuzzy*. Outra situação indesejável dessa função é o fato de não conhecer o valor máximo do intervalo. Esse valor máximo pode ser diferente em diversas situações, pois o mesmo depende da maior e da menor nota do usuário ao avaliar um conjunto de tuplas. Com essa função, os valores encontrados sempre estão em um determinado intervalo $m_{ij} \in [0, \frac{max}{min}]$. No exemplo 3, o intervalo é caracterizado por valores $m_{ij} \in [0, 5]$, mas em outras situações esse limite superior do intervalo pode mudar de acordo com a distribuição das notas que avaliam o conjunto de tuplas.

É necessário portanto, estabelecer um intervalo fixo e conhecido, ou seja, é preciso que todos os valores m_{ij} da matriz de preferência M estejam no intervalo [0,1]. Estes valores, por sua vez, devem satisfazer as propriedades requeridas por uma relação de preferência *fuzzy*. A partir das conclusões obtidas por essa transformação inicial, uma segunda maneira de encontrar os valores da matriz de preferência M é descrita na próxima subseção. Essa segunda maneira para encontrar os valores de uma matriz de preferência pode ser entendida como um refinamento desta primeira transformação. O foco inicial é encontrar uma forma de estabelecer o intervalo [0,1] para todos os valores da matriz e depois garantir que esses valores satisfaçam as propriedades das relações de preferências *fuzzy*.

2.4.2 Refinamento da Obtenção dos Valores da Matriz de Preferência

O objetivo deste refinamento é usar uma função sobre os valores m_{ij} da matriz M encontrada na pela transformação inicial descrita na subseção 2.4.1, a fim de obter uma matriz resultante que satisfaça as propriedades requeridas por uma relação de preferência *fuzzy*.

É necessário utilizar uma função de transformação $f : R \rightarrow [0, 1]$. Essa função deve ser crescente, pois é preciso garantir que os valores m_{ij} da matriz M encontrados por essa segunda transformação cresça na mesma proporção dos valores m_{ij} encontrados na

transformação inicial descrita anteriormente.

Por exemplo, considere os elementos $m_{12} = 1.67$ e $m_{13} = 5$ da matriz M encontrada pela primeira transformação ilustrada na Figura 2.8(b). Ao aplicar uma função para normalizar esses valores no intervalo [0,1], é necessário garantir que m_{12} não obtenha um valor maior do que m_{13}, pois isso seria uma contradição aos valores já encontrados.

Algumas funções bastante utilizadas na literatura são àquelas pertencentes à família de funções baseadas na escala de razão (Tanino, 1984), (Chiclana *et al.*, 1998), (Chiclana *et al.*, 2001) definida por:

$$f_n(x) = \frac{x^n}{1 + x^n} \text{ onde } n = (1, ..., k) \tag{2.1}$$

Aplicando a razão $\frac{u_i}{u_j}$ usada na Seção 2.4.1 na função 2.1, a seguinte função de transformação é obtida:

$$f_n(\frac{u_i}{u_j}) = \frac{(\frac{u_i}{u_j})^n}{1 + (\frac{u_i}{u_j})^n} = \frac{u_i^n}{u_i^n + u_j^n} \tag{2.2}$$

Portanto, a função 2.2 é utilizada para normalizar os valores m_{ij} no intervalo [0,1]. E para qualquer valor de $n = (1, ..., k)$, essa função atende as propriedades requeridas por uma relação de preferência *fuzzy*. Para fixar melhor cada propriedade e verificar se a função definida nesta segunda transformação satisfaz tais propriedades considere o Exemplo 2.3.

Exemplo 2.3. Suponha o mesmo conjunto de tuplas do Exemplo 2.2, ilustrado na Figura 2.8(a) e a matriz de preferência encontrada pela primeira transformação ilustrada na Figura 2.8(b). Aplica-se a função 2.2, tal que $n = 2$, em todos os elementos da matriz M encontrada pela transformação inicial. Ao aplicar essa função nos elementos, a relação de preferência *fuzzy* ilustrada pela seguinte matriz de preferência $M_{5\times5}$ é obtida:

$$M = \begin{bmatrix} \mathbf{0.5} & 0.73 & 0.96 & 0.86 & 0.86 \\ 0.27 & \mathbf{0.5} & 0.86 & 0.69 & 0.69 \\ 0.04 & 0.14 & \mathbf{0.5} & 0.2 & 0.2 \\ 0.14 & 0.31 & 0.8 & \mathbf{0.5} & 0.5 \\ 0.14 & 0.31 & 0.8 & 0.5 & \mathbf{0.5} \end{bmatrix}$$

Essa matriz de preferência M atende as propriedades de uma relação de preferência *fuzzy*. No entanto, é preciso verificar se qualquer matriz de preferência obtida pela função 2.2 atende essas propriedades, para qualquer valor de $n = (1, ..., k)$.

Para realizar a verificação dessas propriedades é preciso relembrar cada uma delas e verificar se são atendidas por esse tipo de função:

1. A primeira propriedade diz que todos os valores m_{ij} de uma matriz M devem estar no intervalo [0,1]. Como os valores da matriz são encontrados pela função 2.2, aplicando essa função na primeira propriedade, tem-se:

$$m_{ij} \in [0,1] \Rightarrow \frac{u_i^n}{u_i^n + u_j^n} \in [0,1]$$

Essa propriedade sempre será atendida pela função 2.2, pois o denominador $(u_i^n + u_j^n)$ sempre terá valor maior ou igual ao numerador (u_i^n). O denominador terá valor igual ao numerador somente quando $u_j = 0$, isto é, quando a nota de t_j for 0. Caso contrário, o valor do denominador sempre será maior que o valor do numerador. Dessa forma, garante-se que o valor m_{ij} estará no intervalo [0,1].

2. A segunda propriedade indica, que em uma matriz de preferência M, os valores m_{ij} serão maiores que 0.5, se e somente se, a nota de t_i for maior que a nota de t_j. Logo, esta segunda propriedade também é garantida, pois aplicando a função 2.2 pode-se observar:

$$E : m_{ij} = \frac{u_i^n}{u_i^n + u_j^n}$$
$$m_{ij} > \frac{1}{2} \Leftrightarrow E > \frac{1}{2}$$
$$\text{Mas, } E > \frac{1}{2} \Leftrightarrow u_i^n > u_j^n$$
$$\text{Logo, } m_{ij} > \frac{1}{2} \Leftrightarrow u_i^n > u_j^n$$

Olhando as relações estabelecidas, percebe-se que, se $u_i^n > u_j^n$, então obviamente $u_i > u_j$. Por exemplo, suponha as notas $u_i = 4$ e $u_j = 2$. Aplicando a função 2.2, tal que $n = 2$, o valor do grau de preferência $m_{ij} = 0.8$ é obtido e este valor atende essa segunda propriedade.

3. A terceira propriedade indica reciprocidade, isto é, o valor m_{ji} (*grau de preferência* tupla t_i sobre a tupla t_j), de modo que a soma dos dois valores seja 1, isto é, o valor máximo do intervalo [0,1]. Ao aplicar a função 2.2 na terceira propriedade, é notável a seguinte relação:

$$m_{ij} + m_{ji} \Rightarrow \frac{u_i^n}{u_i^n + u_j^n} + \frac{u_j^n}{u_j^n + u_i^n} \Rightarrow \frac{u_i^n + u_j^n}{u_i^n + u_j^n} = 1$$

Logo, essa terceira propriedade também é garantida por valores m_{ij} encontrados pela função de transformação 2.2.

4. A quarta propriedade indica indiferença entre as tuplas e é representada em uma matriz de preferência M pelo valor $m_{ij} = 0.5$. Essa quarta propriedade também é garantida pela função 2.2, pois, aplicando essa função observa-se a seguinte relação:

$$E : m_{ij} = \frac{u_i^n}{u_i^n + u_j^n}$$
$$m_{ij} = \frac{1}{2} \Leftrightarrow E = \frac{1}{2}$$
$$\text{Mas, } E = \frac{1}{2} \Leftrightarrow u_i^n = u_j^n$$
$$\text{Logo, } m_{ij} = \frac{1}{2} \Leftrightarrow u_i^n = u_j^n$$

Nesta quarta propriedade, o valor que indica indiferença é encontrado somente se a nota de de uma tupla t_i for igual à nota de outro tupla t_j. Assim, é notável que para qualquer valor $u_i = u_j$ essa quarta propriedade é atendida por esse tipo de função, pois o valor do denominador sempre será o dobro do valor existente no numerador.

Essas propriedades das relações de preferências *fuzzy* são atendidas por matrizes de preferências, cujos valores m_{ij} são encontrados por funções do tipo $f_n(\frac{u_i}{u_j}) = \frac{u_i^n}{u_i^n + u_j^n}$, onde $n = (1, ..., k)$. Nesta pesquisa, é usada esse tipo de função para obter as matrizes de preferências representando as avaliações do usuário sobre um conjunto de *tuplas*.

2.5 Modelos de Preferências

Uma vez estabelecido os conceitos necessários para entendimento das diversas representações de preferências *quantitativas* do usuário, bem como, funções que transformam notas de tuplas em relações de preferências *fuzzy*. Tem-se a necessidade de apresentar os modelos de preferências *qualitativos*, seus respectivos conceitos e as principais diferenças entre eles.

Rigorosamente falando, os modelos de preferências *qualitativos* podem ser divididos em duas categorias: modelos de preferências não condicionais e modelos de preferências condicionais. Esta seção está dividida em duas subseções para compreender melhor as características de cada um desses modelos de preferências.

2.5.1 Modelos de Preferências Não Condicionais

Modelos de Preferências Não Condicionais se caracterizam pelo fato das preferências sobre valores dos atributos não dependerem de valores de outros atributos. Um dos modelos mais comuns nesta linha é o Modelo de Preferência Pareto. Uma preferência Pareto é definida por:

Definição 2.6. (Preferência Pareto) Seja $R(A_1, ..., A_n)$ um esquema relacional e $Tup(R) = \{t_1, ..., t_k\}$ um conjunto de tuplas sobre o esquema relacional R. Uma preferência Pareto $t \succ_{\otimes} t'$ sobre duas tuplas é constatada se e somente se:

- Existe $a_i \in A_i$, tal que, $(t_{a_i} \succ_{A_i} t'_{a_i})$, e
- Para todo $A_j \neq A_i$: $(t_{a_j} \succ_{A_j} t'_{a_j})$ ou $(t_{a_j} = t'_{a_j})$

De modo geral, para uma determinada tupla t ser preferível a outra tupla t', é necessário que ele seja preferível em pelo menos um dos atributos e nos demais atributos ele deve ser também preferível ou tenha o mesmo valor, isto é, seja igualmente preferido. Para ilustrar essa definição considere um exemplo prático.

Exemplo 2.4. Suponha um esquema relacional de carros com os seguintes atributos Carros(Ano,Km,Preço). Suponha um determinado usuário "y"que está pesquisando carros para realizar uma compra. Este usuário define algumas prioridades sobre as características do carro que está procurando: (1) prefere carros mais novos; (2) prefere os carros com menos quilômetros rodados e (3) prefere os carros com menor valor de preço. A partir dessas prioridades do usuário é possível denotar as regras de preferências ilustradas na Tabela 2.1.

Tabela 2.1: Regras de Preferências

r_1	$\text{Ano}_{(maior)} \succ \text{Ano}_{(menor)}$
r_2	$\text{km}_{(menor)} \succ \text{km}_{(maior)}$
r_3	$\text{Preço}_{(menor)} \succ \text{Preço}_{(maior)}$

Considere para este exemplo o conjunto de tuplas sobre o esquema relacional de carros ilustrado pela Tabela 2.2.

Tabela 2.2: Lista de Carros

Id	**Ano**	**km**	**Preço**
t_1	2009	60.000	27.000
t_2	2008	80.000	25.700
t_3	2012	30.000	32.000
t_4	2012	30.000	27.000

Para estabelecer que a tupla t_1 é preferível a tupla t_2, é necessário que em um dos atributos, a tupla t_1 seja preferível a t_2 e para os demais atributos ela seja preferível ou igualmente preferível a t_2. Analisando as regras de preferências ilustradas na Tabela 2.1 pode-se concluir que as tuplas t_1 e t_2 são incomparáveis pela preferência Pareto. Nos atributos "Ano"e "km", a tupla t_1 é preferível a t_2, pois $\text{Ano}(t_1) = 2009 > \text{Ano}(t_2) = 2008$ e $\text{km}(t_1) = 60.000 < \text{km}(t_2) = 80.000$. No entanto, para que $t_1 \succ_{\otimes} t_2$ é necessário que no atributo "Preço", a tupla t_1 seja preferível ou igualmente preferível a tupla t_2, o que não acontece, pois $\text{Preço}(t_1) = 27.000 > \text{Preço}(t_2) = 25.700$, contradizendo a regra r_3 e consequentemente a definição de preferência Pareto.

Agora, considere a comparação das tuplas t_1 e t_4. Analisando as mesmas regras de preferências, pode-se concluir que t_4 é preferível a t_1 pela preferência Pareto. Observando os valores dos três atributos de cada tupla é possível notar que nos atributos "Ano"e "km"a tupla t_4 é preferível a tupla t_1, pois $\text{Ano}(t_1) = 2012 > \text{Ano}(t_4) = 2009$ e $\text{km}(t_1) = 30.000 < \text{km}(t_4) = 60.000$. E no terceiro atributo, as duas tuplas são igualmente preferidas, pois, $\text{Preço}(t_1) = 27.000$ é o mesmo valor do $\text{Preço}(t_4) = 27.000$. Ou seja, não existe outro valor de atributo que indica que a tupla t_1 é preferível a tupla t_4, logo, pela definição de preferência Pareto pode-se concluir que $t_4 \succ_{\otimes} t_1$.

O conceito de preferência Pareto é um conceito forte e existe certa dificuldade ao comparar duas tuplas, devido às preferências que devem ser respeitadas em cada atributo.

Na próxima subseção é apresentado o modelo de preferência condicional, que de um modo geral é mais expressivo que o modelo de preferência Pareto.

2.5.2 Modelos de Preferências Condicionais

Modelos de Preferências Condicionais ou Contextuais se caracterizam pelo fato dos valores de um determinado atributo influenciar na escolha de valores em outros atributos. Diferente do Modelo de Preferência Pareto, as preferências deste modelo não são individuais para cada atributo. O usuário pode preferir um determinado valor para um atributo dependendo do valor que outro atributo assuma.

Com o intuito de representar este tipo de preferência, Wilson (2004) apresentou uma definição formal denominada *regra de preferência contextual* definida por:

Definição 2.7. (Regra de Preferência Contextual) Uma regra de preferência contextual (*cp-rule*), consiste em uma declaração na forma: $\varphi : u \rightarrow (X = x) \succ (X = x')[W]$. Onde, u é uma instrução do tipo $(A_{i_1} = a_{i_1}) \wedge \ldots \wedge (A_{i_n} = a_{i_n})$ que é chamada condição ou contexto da regra, $(X = x) \succ (X = x')$ é a preferência e $[W]$ corresponde ao conjunto de atributos cujo valores são desconsiderados pela *cp-rule*. Um conjunto finito de regras de preferências contextuais forma uma teoria de preferências contextuais (*cp-teoria*).

Para ilustrar a essência da comparação de tuplas por regras de preferências contextuais considere o Exemplo 2.5.

Exemplo 2.5. Suponha o esquema relacional Filme(Título,Gênero,Ator,Diretor). Considere o conjunto de filmes $Tup = \{t_1, t_2, t_3, t_4, t_5\}$ sobre este esquema relacional ilustrado pela Tabela 2.3.

Tabela 2.3: Filmes

	TÍTULO (T)	GÊNERO (G)	ATOR (A)	DIRETOR (D)
t_1	A lista de Schindler	Drama	Ben Kingsley	Steven Spielberg
t_2	Tudo pode dar certo	Comédia	Larry David	Woody Allen
t_3	O resgate do soldado Ryan	Ação	Tom Hanks	Steven Spielberg
t_4	A procura da Felicidade	Drama	Will Smith	Grabriele Muccino
t_5	Náufrago	Drama	Tom Hanks	Robert Zemeckis

Suponha que um usuário "y"especifique suas preferências pela *cp-teoria* $\Gamma = \{\varphi_1, \varphi_2, \varphi_3\}$, onde:

- φ_1 : (G=Comédia) $\succ_G$ (G=Drama)[T,A,D]
- φ_2 : D=Steven Spielberg $\rightarrow$ (G=Ação) $\succ_G$ (G=Drama)[T,A]
- φ_3 : G=Drama $\rightarrow$ (A=Tom Hanks) $\succ_A$ (A=Will Smith)[T,D].

Para os filmes ilustrados na Tabela 2.3, a *cp-rule* φ_1 infere que o filme t_2 é preferível aos filmes t_1, t_4 e t_5, pois essa *cp-rule* só depende do atributo gênero para realizar a

comparação de tuplas. Os demais atributos estão entre chaves, ou seja, os valores assumidos por estes atributos não influenciam a preferência do usuário. Logo, uma regra de preferência contextual não precisa necessariamente ter um contexto.

A *cp-rule* φ_2 infere que o filme $t_3 \succ t_1$, pois, o contexto de ambos os filmes é "D=Steven Spielberg", e, como os atributos "T"e "A"são desconsiderados, para inferir a preferência olha-se apenas para o atributo "G"cujo usuário definiu que prefere "Ação"a "Drama". Já pela *cp-rule* φ_2 é possível inferir que o filme $t_5 \succ t_4$.

Diferente da preferência Pareto, nas regras de preferências contextuais não existem preferências individuais dos atributos interferindo na decisão da preferência entre duas tuplas. Isso é considerado uma vantagem do Modelo de Preferência Contextual sobre o Modelo Pareto, pois através das regras de preferências contextuais é possível comparar um maior número de tuplas. Portanto, esta pesquisa se enquadra neste tipo de modelo de preferência.

2.6 Noções de Consistência em Relações de Preferências Fuzzy

Noções de consistência em relações de preferências *fuzzy* é um campo de pesquisa que tem sido muito estudado entre os pesquisadores da área de Preferências *Fuzzy*. A ideia de consistência nessas relações está ligada a propriedade de transitividade, que segundo Herrera-Viedma *et al.* (2004) é uma das principais e mais importantes propriedades relativa à preferências.

Herrera-Viedma *et al.* (2004) explica diversas propriedades de transitividade em relações de preferências *fuzzy*, tais como: Condição Triângulo, Transitividade Fraca, Transitividade Max-Min, Transitividade Max-Max, Transitividade Max-Min Restrita, Transitividade Max-Max Restrita e Transitividade Aditiva.

Embora exista, essa diversidade de propriedades relacionadas a consistência de relações de preferências *fuzzy*, as propriedades mais comuns e estudadas na literatura são: Transitividade Fraca e Transitividade Aditiva. A seguir os conceitos de cada uma dessas propriedades são apresentados.

Para compreender melhor cada uma das propriedades descritas nas próximas subseções, suponha uma matriz de preferência $M_{n\times n}$ e suponha ainda que os *graus de preferências* que preenchem essa matriz foram obtidos por uma função de transformação $f_n(\frac{u_i}{u_j}) = \frac{u_i^n}{u_i^n+u_j^n}$, como definida na Seção 2.4 do Capítulo 2.

Para as duas propriedades apresentadas nas próximas subseções, considere três tuplas distintas t_i, t_j e t_k, representando tuplas avaliadas pelo usuário e os graus de preferências m_{ij} (representando o grau de preferência da tupla t_i sobre t_j), m_{jk} (representando o grau de preferência de t_j sobre t_k) e m_{ik} (representando o grau de preferência de t_i sobre t_k).

2.6.1 Transitividade Fraca

A definição de transitividade fraca para relações de preferências *fuzzy* é conceituada pela seguinte propriedade:

$$m_{ij} \geq \frac{1}{2}, m_{jk} \geq \frac{1}{2} \Rightarrow m_{ik} \geq \frac{1}{2} \ \forall i, j, k \text{ da matriz de preferência} \tag{2.3}$$

Esse tipo de transitividade é a propriedade mínima que uma relação de preferência *fuzzy* deve atender para ser considerada consistente. Ela é interpretada como uma regra do tipo "SE ... ENTÃO": Se uma tupla t_i é preferível a uma tupla t_j e esta preferível a tupla t_k, então a tupla t_i também é preferível a tupla t_k. Portanto, se uma relação de preferência *fuzzy* garante essa propriedade ela é dita minimamente consistente.

Nessa propriedade, sempre que o antecedente da fórmula é verdadeiro, o consequente também é verdadeiro. Nesta pesquisa, os valores m_{ij} de uma matriz de preferência *fuzzy* são obtidos por alguma função de transformação e uma relação de preferência *fuzzy* obtida pela família de funções usadas nessa dissertação, sempre garante essa propriedade de transitividade fraca.

Prova. Prova dessa propriedade para a função $f_n(\frac{u_i}{u_j}) = \frac{u_i^n}{u_i^n+u_j^n}$

A propriedade de transitividade fraca é definida pela propriedade 2.3. Os valores m_{ij} da matriz de preferência são obtidos pela função de transformação $f_n(\frac{u_i}{u_j}) = \frac{u_i^n}{u_i^n+u_j^n}$ e os valores obtidos por essa função são representados por uma matriz na seguinte forma:

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{bmatrix}$$

Para o antecedente da propriedade (1), isto é, $m_{ij} \geq \frac{1}{2}$ e $m_{jk} \geq \frac{1}{2}$ tem-se as seguintes relações:

(2) $m_{ij} \geq \frac{1}{2} \Rightarrow \frac{u_i^n}{u_i^n+u_j^n} \geq \frac{1}{2} \Rightarrow 2u_i^n \geq u_i^n + u_j^n \Rightarrow u_i^n \geq u_j^n$

(3) $m_{jk} \geq \frac{1}{2} \Rightarrow \frac{u_j^n}{u_j^n+u_k^n} \geq \frac{1}{2} \Rightarrow 2u_j^n \geq u_j^n + u_k^n \Rightarrow u_j^n \geq u_k^n$

A partir das relações (2) e (3), é possível inferir que $u_i^n \geq u_k^n$, isto é, a relação resultante do consequente da propriedade (1):

(4) $m_{ik} \geq \frac{1}{2} \Rightarrow \frac{u_i^n}{u_i^n+u_k^n} \geq \frac{1}{2} \Rightarrow 2u_i^n \geq u_i^n + u_k^n \Rightarrow u_i^n \geq u_k^n$

Conhecendo a relação expressa em (4), isto é, $u_i^n \geq u_k^n$ e sabendo que o grau de preferência m_{ik} é obtido pela função $f_n(\frac{u_i}{u_j}) = \frac{u_i^n}{u_i^n+u_j^n}$, tem-se um caso base para a função f_n quando $u_i^n = u_k^n$ e que resultará em:

$$m_{ik} = \frac{u_i^n}{u_i^n+u_k^n} = \frac{u_i^n}{u_i^n+u_i^n} = \frac{u_i^n}{2u_i^n} = \frac{1}{2}$$

O valor resultante dessa função sempre assumirá valores maiores ou iguais a $\frac{1}{2}$ quando as relações expressas em (2),(3) e (4) forem verdadeiras, e isso pode ser expressa pela propriedade: $m_{ij} \geq \frac{1}{2}, m_{jk} \geq \frac{1}{2} \Rightarrow m_{ik} \geq \frac{1}{2} \forall i, j, k$ da matriz de preferência.

2.6.2 Transitividade Aditiva

A propriedade de Transitividade Aditiva juntamente com a Transitividade Fraca são as propriedades mais comuns e vistas na literatura. No entanto, ao contrário da Transitividade Fraca, a Transitividade Aditiva é um conceito mais forte e difícil de obter na prática. Essa propriedade requer que a relação de preferência *fuzzy* possua transitividade fraca e, além disso, que os elementos da relação satisfaçam valores estritos. Isso nem sempre é possível, pois as preferências e/ou julgamentos dos seres humanos raramente estão em conformidade com fórmulas matemáticas.

A formulação matemática dessa propriedade foi dada primeiramente por Tanino (1984), e tem a seguinte definição:

$$(m_{ij} - 0.5) + (m_{jk} - 0.5) = (m_{ik} - 0.5) \tag{2.4}$$

Esse tipo de transitividade tem a seguinte interpretação: Suponha que é preciso estabelecer um *ranking* entre três tuplas distintas e inicialmente não se tem informação alguma sobre as preferências entre essas três tuplas. Logo, pode-se dizer que inicialmente existe uma situação de indiferença, isto é, $t_i \sim t_j \sim t_k$. Em termos de *graus de preferências* essa situação é representada por $m_{ij} = m_{jk} = m_{ik} = 0.5$.

Suponha agora, que existe um pouco mais de informação a respeito das preferências entre as tuplas e essa informação diz que $t_j \succ t_i$ (t_j é preferível a t_i), isto é, o *grau de preferência* de t_i sobre t_j é dado por um valor $m_{ij} < 0.5$. Com essa nova informação, nota-se que um dos dois valores m_{jk} ou m_{ik} deve mudar, isto é, um desses dois valores deve ser diferente de 0.5. Caso contrário, haveria uma contradição, pois existiria a seguinte situação: $t_j \sim t_k$, $t_k \sim t_i$, e, por transitividade, $t_j \sim t_i$. O que não é verdade, pois já existe a informação que $t_j \succ t_i$ com um *grau de preferência* $m_{ij} < 0.5$.

Suponha então, que $m_{jk} = 0.5$. Logo, existe a seguinte situação até o momento: t_j é preferível a t_i e não existe preferência entre t_j e t_k, isto é, eles são considerados indiferentes. Sendo assim, deve-se concluir que t_k deve ser preferível t_i, pois se fosse o contrário, $t_i \succ t_k$ e como já existe a preferência $t_j \succ t_i$, por transitividade a conclusão seria $t_j \succ t_k$, o que é uma contradição, pois já se tem a informação de que $t_j \sim t_k$. Além disso, como $t_j \sim t_k$, então $m_{ij} = m_{ik}$ e assim $(m_{ij} - 0.5) + (m_{jk} - 0.5) = (m_{ij} - 0.5) = (m_{ik} - 0.5)$.

Por exemplo, suponha os seguintes valores de graus de preferências $m_{ij} = 0.4$ e $m_{jk} = 0.5$, isto é: (1) $t_j \succ t_i$ (t_j é preferível a t_i) (2) $t_j \sim t_k$ (t_j e t_k são indiferentes)

Logo, t_k deve ser preferível a t_i para que não haja contradição nesta situação e como $t_j \sim t_k$, então $m_{ij} = m_{ik} = 0.4$. Sendo assim, tem-se:

$$(m_{ij} - 0.5) + (m_{jk} - 0.5) = (m_{ik} - 0.5) \Rightarrow$$
$$(0.4 - 0.5) + (0.5 - 0.5) = (0.4 - 0.5) \Rightarrow -0.1 = -0.1$$

Como já foi mencionado, essa propriedade é bastante difícil de garantir na prática, pois nem sempre os usuários especificam exatamente os valores de *graus de preferências* como especificados matematicamente por essa propriedade.

2.6.3 Discussão sobre as Propriedades

A propriedade de transitividade fraca apresentada na subseção 2.6.1, é a propriedade mais comum e mais utilizada na literatura (Xu *et al.*, 2013), (Ma *et al.*, 2006) para avaliar a consistência de uma relação de preferência *fuzzy*, principalmente quando esta relação é fornecida por um usuário, uma vez que, a propriedade de transitividade aditiva é muito forte e difícil de ser garantida.

A propriedade de transitividade aditiva geralmente é utilizada para auxiliar no processo de reparação de inconsistência de uma relação de preferência *fuzzy* (Ma *et al.*, 2006). No entanto, a avaliação da consistência e/ou inconsistência da relação é feita com base na transitividade fraca. Portanto, nesta pesquisa, as relações de preferências *fuzzy* inferidas pelo modelo de preferência proposto são consideradas consistentes se atendem a propriedade de transitividade fraca.

2.7 Considerações do Capítulo

Neste capítulo foram introduzidos os principais conceitos para compreender o contexto do problema de mineração tratado nesta dissertação. Num primeiro momento foi apresentado como é tratado o problema geral de mineração de preferências no qual este trabalho se enquadra. Após isso, se fez necessário apresentar os diversos tipos de representação quantitativas de preferências (*crisp* e *fuzzy*), bem como, a transformação de representação de preferência *crisp* em representação de preferência *fuzzy*. Por fim, foram apresentados os modelos de preferências mais comuns encontrados nos trabalhos relacionados ao tópico de mineração de preferências, bem como, noções de consistência em relações de preferências *fuzzy* que servem de base para os métodos de reparação de inconsistência destacados nos trabalhos relacionados ao tópico de métodos de reparação de inconsistência.

Capítulo 3

Trabalhos Relacionados

Este capítulo tem como principal objetivo destacar os trabalhos relacionados às duas sub-áreas no qual essa pesquisa se enquadra. Portanto, a Seção 3.1 destaca os principais trabalhos relacionados ao tópico de mineração de preferências e a Seção 3.2 aborda alguns trabalhos relacionados ao tópico de Métodos de Reparação de Inconsistência.

3.1 Mineração de Preferências

Técnicas de Mineração de Preferências envolvem diversas formas de aprender preferências, tais formas podem ser *quantitativas* ou *qualitativas* (Stefanidis *et al.*, 2011). Em Fürnkranz e Hüllermeier (2011) é apresentando um extensivo texto sobre técnicas e abordagens para aprendizagem de preferências. De um modo geral, a aprendizagem de preferência pode ser dividida em dois subproblemas distintos: *Label Ranking* e *Object Ranking*. O problema de *Label Ranking* visa predizer uma ordem sobre um conjunto de *labels* de um determinado objeto (tupla) (Hüllermeier *et al.*, 2008), (de Sá *et al.*, 2011) e *Object Ranking* visa predizer qual o objeto (tupla) preferido quando se compara dois objetos (tuplas). Esta pesquisa se enquadra neste último tipo de problema, portanto nesta seção serão abordados trabalhos relacionados à *object ranking*.

Holland *et al.* (2003) propõe um método para minerar preferências do usuário a partir de um *log* de dados gerado por um servidor, quando o usuário acessa um *site*. O trabalho possui o Modelo de Preferência Pareto subjacente e consiste em obter preferências a partir de um arquivo de *log* e retornar a preferência do usuário sobre as tuplas. Uma tupla é considerada preferida se a mesma aparece com muita frequência no arquivo de *log*. As preferências preditas por esse modelo não expressam preferências condicionais e

não utilizam representação *fuzzy*.

A abordagem de aprendizagem de preferências apresentado em Jiang *et al.* (2008), consiste em um método de Mineração de Preferências que utiliza amostras de preferências fornecidas pelo usuário, para inferir um ordem sobre qualquer par de tuplas do banco de dados. Essas amostras são divididas em amostras superiores e inferiores, e contém informações sobre quais tuplas o usuário mais gosta e quais ele menos gosta, respectivamente. Este trabalho também não usa representação de preferências *fuzzy* e não tem o modelo de preferência contextual subjacente.

Concentrando no tópico de mineração de regras de preferências contextuais Koriche e Zanuttini (2010) propõe um algoritmo para minerar o modelo CP-net (Boutilier *et al.*, 2004) a partir de um conjunto de preferências fornecido pelo usuário. As amostras de preferências são representadas por um conjunto de pares de tuplas ordenadas. O objetivo do trabalho é identificar uma ordenação de preferência com uma CP-Net, através da iteração com o usuário a partir de um pequeno número de consultas.

Outro trabalho relacionado ao tópico de mineração de regras de preferências contextuais é proposto em de Amo *et al.* (2012b). Neste trabalho, os autores apresentam um método para encontrar um perfil do usuário a partir de amostras de suas preferências. O perfil do usuário é especificado por um conjunto de regras de preferências contextuais. O método proposto neste trabalho é dividido em dois algoritmos: o primeiro algoritmo nomeado *ContPrefMiner* é responsável pela mineração de regras de preferências contextuais, e o segundo algoritmo nomeado *ProfMiner* encontra o perfil do usuário a partir das regras de preferências fornecidas pelo *ContPrefMiner*. Este trabalho tem o modelo de preferência contextual subjacente, mas não usa representação de preferência *fuzzy*.

O trabalho apresentado em de Amo *et al.* (2012a) apresenta um algoritmo baseado em redes *bayesianas*. Este trabalho é o principal *baseline* de comparação desta pesquisa. Nele é apresentado um método qualitativo para minerar regras de preferências contextuais a partir de escolhas passadas do usuário. O método nomeado como *CPrefMiner* recebe como entrada, um conjunto de pares de tuplas e a partir deste descobre um grafo e um conjunto de tabelas de probabilidades condicionais, expressando as preferências do usuário. Este trabalho incentivou o início desta pesquisa, uma vez que, ele não utiliza representação *fuzzy* para expressar as preferências do usuário. Rigorosamente falando, a representação *fuzzy* é mais rica, pois é possível estimar *o quanto* o usuário prefere uma tupla em relação a outra.

Com relação à mineração de preferências a partir de relações de preferências *fuzzy*, não foi encontrado até o momento trabalhos que utilizam este tipo particular representação de preferências do usuário em um ambiente real, e a partir delas descobrem um modelo que possa predizer a preferência sobre novas tuplas.

A maioria dos trabalhos que utilizam preferência *fuzzy* estão relacionados ao tópico de sistemas de recomendações. Os trabalhos que usam preferências *fuzzy* em sistemas

de recomendação, ressaltam que este tipo de representação é mais rica e melhoram a qualidade recomendação. A seguir, são apresentados dois trabalhos que utilizam conjuntos *fuzzy* para representar preferências do usuário.

Em Eckhardt e Vojtás (2009) é apresentado um algoritmo para recomendar objetos (*tuplas*) aos usuários. Para fazer a recomendação, o modelo se baseia na ideia de dependência dos atributos da CP-Net (Boutilier *et al.*, 2004) (mas não utilizam regras de preferências contextuais) e na teoria dos conjuntos *fuzzy* para representar as preferências. Basicamente, um usuário avalia um conjunto pequeno de tuplas com notas variando no intervalo [1,...,5]. Desse conjunto, o método constrói uma pequena CP-Net, onde, atributos numéricos dependem de atributos nominais.

Com a CP-Net construída, o modelo é dividido em duas fases: a primeira fase consiste em encontrar os valores *fuzzy* de cada valor de atributo da tupla . A segunda fase utiliza uma função de agregação para avaliar uma tupla. Dessa forma, se os valores de todos os atributos de uma tupla t_1 são menores que todos os valores de uma tupla t_2, o modelo conclui que a tupla t_2 é preferível a tupla t_1.

Após encontrar o valor *fuzzy* da tupla pela função de agregação é gerado um *ranking* de tuplas preferidas, ou seja, tuplas com maiores valores ficam no topo. O algoritmo então recomenda a primeira tupla do *ranking*, isto é, a mais preferida. Este trabalho utiliza uma abordagem totalmente diferente da proposta desta dissertação, pois encontra o valor *fuzzy* para cada valor de atributo e só após encontra o grau de preferência da tupla. Além disso, não trata questões de inconsistência como é proposta nesta dissertação.

Outro trabalho que utiliza conjuntos *fuzzy* para auxiliar algoritmos de sistemas de recomendação é visto em Zenebe *et al.* (2010). Este trabalho, representa preferências do usuário com base apenas em um atributo "X"multivalorado da tupla. Um primeiro algoritmo que determina o valor *fuzzy* dos vários valores do atributo "X"é proposto. Este algoritmo, recebe como entrada um conjunto de *tuplas* avaliadas pelo usuário e retorna quatro vetores de preferências: Preferido(PX), Não Preferido (NX), Indiferente (IX) e Desconhecido (UX).

Após encontrar os quatro vetores, o segundo algoritmo é executado para descobrir a preferência sobre uma nova tupla. O segundo algoritmo, recebe uma nova tupla e usa uma medida de similaridade para descobrir a proximidade entre a tupla e os vetores retornados pelo primeiro algoritmo e o vetor mais similar classifica a nova tupla. Dessa forma, uma tupla pode ser preferida (P), não preferida (NP), indiferente (I) ou desconhecida (D). A tupla é recomendada se a classe inferida for preferida (P), ou seja, se o vetor PX for mais similar a nova tupla. Este método apresentado por Zenebe *et al.* (2010) não leva em consideração todos os atributos de uma tupla para aprender as preferências e também, não compara duas *tuplas* como é proposto nesta dissertação.

A Tabela 3.1 resume a comparação dos trabalhos destacados nesta seção.

Tabela 3.1: Comparação dos Trabalhos Correlatos

TRABALHO	**TIPO DE REPRESENTAÇÃO**	**ENTRADA**	**SAÍDA**	**MODELO DE PREFERÊNCIA**	**EXPERIMENTOS**
Holland *et al.* (2003)	*Crisp*	Arquivo de *Log* (*Ranking* de Tuplas)	Conjunto de preferências	Pareto	Dados Sintéticos e Dados Reais
Jiang *et al.* (2008)	*Crisp*	Um conjunto O de *tuplas*; um conjunto S de *tuplas* preferidas; um conjunto Q de *tuplas* não preferidas	Conjunto de preferências satisfazendo S e Q	Pareto	Dados Sintéticos e Dados Reais
Koriche e Zanuttini (2010)	*Crisp*	Conjunto de pares de tuplas	CP-Net	Contextual	Dados Sintéticos
de Amo *et al.* (2012b)	*Crisp*	Conjunto de pares de tuplas	Conjunto de regras de preferências contextuais	Contextual	Dados Reais
de Amo *et al.* (2012a)	*Crisp*	Conjunto de pares de tuplas	Rede de Preferências	Contextual	Dados Sintéticos e Dados Reais
Eckhardt e Vojtás (2009)	*Fuzzy*	Conjunto de tuplas avaliadas pelo usuário	Sistema de Recomendação	Condicional (mas não utilizam *cp-rules*)	Dados Reais
Zenebe *et al.* (2010)	*Fuzzy*	Conjunto de tuplas avaliadas pelo usuário	Sistema de Recomendação	-	Dados Reais
Este Trabalho	*Fuzzy*	Conjunto de Pares de *Tuplas Fuzzy*	Rede de Preferências	Contextual	Dados Reais

3.2 Métodos de Reparação de Inconsistência

Noções de consistência em relações de preferências fuzzy estão diretamente ligadas à propriedade de transitividade em preferências. Em Herrera-Viedma *et al.* (2004) são apresentadas diversas noções de consistência em relações de preferências *fuzzy* e apresentam diversas propriedades para analisar a consistência de uma relação de preferência *fuzzy*, tais como: transitividade fraca, transitividade aditiva, transitividade max-min, transititivade max-max.

O trabalho Ma *et al.* (2006) investiga problemas de inconsistência de relações de preferências *fuzzy* e apresenta um método baseado em teoria dos grafos para identificar se uma relação de preferência *fuzzy* é consistente ou não. Essa verificação se baseia na propriedade de transitividade fraca. Depois, os autores apresentam um algoritmo com base na propriedade de transitividade aditiva para reparar a inconsistência de uma relação de preferência *fuzzy* e torná-la minimamente consistente.

Em Xu *et al.* (2013) é proposto um método para reparar toda a inconsistência de uma relação de preferência *fuzzy*. O método baseado em transitividade fraca consiste em dois algoritmos: O primeiro algoritmo se baseia em teoria dos grafos para encontrar as inconsistências de uma relação de preferência *fuzzy*. Já o segundo algoritmo, trata-se de um algoritmo recursivo que tem como objetivo eliminar todas as inconsistências da relação de preferência *fuzzy* tornando-a consistente.

Os autores do trabalho Liu *et al.* (2012) discutem diversas propriedades das relações de preferências *fuzzy* e apresentam um método para reparação da inconsistência das relações de preferências *fuzzy* com base na propriedade de transitividade aditiva. Além disso, o método também é utilizado para completar valores em falta em uma relação de preferência *fuzzy* e consequemente reparar a inconsistência da mesma.

O trabalho de Zhang *et al.* (2012) apresenta um método para reparação de inconsistência de uma relação de preferência *fuzzy* com base em modelos de otimização linear. Os modelos propostos no artigo tem como objetivo reparar a inconsistência de uma relação de preferência do usuário, obter um modelo consenso das relações de preferências de vários usuários e trabalhar com relações de preferências *fuzzy* com valores em falta.

Ambos os trabalhos apresentados nesta seção, trata-se de trabalhos teóricos envolvendo este tipo particular de representação de preferências do usuário. Não encontramos até o momento alguma aplicação destes métodos no cenários real de mineração de preferências. Escolhemos o método apresentado em Xu *et al.* (2013) como baseline de comparação dos métodos propostos neste trabalho de dissertação.

3.3 Considerações do Capítulo

Neste Capítulo foram apresentados os principais trabalhos relacionados à esta pesquisa sob dois aspectos: Algoritmos de Mineração de Preferências e Métodos de Reparação de Inconsistência. Primeiramente, foram apresentados os algoritmos de mineração de preferências enfatizando e caracterizando cada trabalho de acordo com os dados utilizados e o modelo de preferência minerado. Num segundo momento, foi apresentado os métodos de reparação de inconsistência em relações de preferências *fuzzy*, envolvendo propriedades de transitividade dessas relações.

Capítulo

4

Background: Mineração de Preferências Contextuais Crisp

Este capítulo tem como principais objetivos: (1) apresentar o problema de mineração de preferências contextuais no cenário *Crisp* introduzido por de Amo *et al.* (2012a) e (2) mostrar com mais detalhes o algoritmo CPrefMiner (de Amo *et al.*, 2012a), (de Amo *et al.*, 2013) proposto para solucionar este problema.

4.1 Formalização do Problema de Mineração no Cenário *Crisp*

Como definido no Capítulo 2, um par (u, v) em um BDP-C (Definição 2.1) representa o fato que o usuário prefere a tupla u a tupla v ($u \succ v$). No cenário *Crisp*, o usuário tem apenas duas opções para as tuplas u e v: ou ($u \succ v$) ou ($v \succ u$).

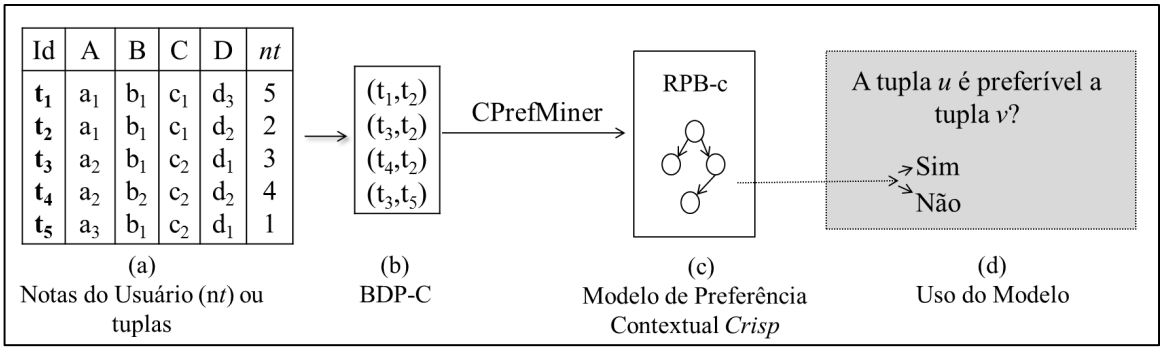


Figura 4.1: Processo da Mineração de Preferências Contextuais no cenário *Crisp*

A Figura 4.1(b) ilustra um banco de dados de preferência *crisp*, representando amostras de alternativas extraídas a partir de informações originais fornecidas pelo usuário sobre suas preferências, ou seja, o banco de dados de preferência é extraído de um conjunto de tuplas avaliadas pelo usuário por notas conforme ilustrado na Figura 4.1(a).

O problema de mineração de preferências contextuais *crisp*, como ilustrado na Figura 4.1, consiste em extrair uma modelo de preferência *crisp* (Figura 4.1(c)) a partir de um banco de dados de preferência *crisp* (Figura 4.1(b)) fornecido pelo usuário. Um modelo de preferência *crisp* tem como objetivo predizer, dado duas novas tuplas v e v, qual a tupla preferida. Neste processo, um modelo de preferência é especificado por uma Rede de Preferência *Bayesiana* definida por:

Definição 4.1. (Rede de Preferências Bayesianas (RPB)) Uma RPB sobre um esquema relacional $R = (A_1, ..., A_n)$ é um par (G, Θ), onde:

1. G é um grafo direcionado acíclico, onde os nós são atributos do esquema relacional e as arestas representam dependências entre os atributos.
2. Θ é uma mapeamento que associa a cada nó de G uma tabela de probabilidade condicional da forma $P[E_2|E_1]$, onde E_1 é um evento do tipo $(A_{i_1} = a_{i_1}) \wedge ... \wedge (A_{i_k} = a_{i_k})$, tal que, $\forall j \in \{1, ..., k\}$ $a_{i_j} \in$ dom(A_{i_j}). E_2 é um evento do tipo "$(B = b_1)$ é preferível a $(B = b_2)$", onde B é um atributo em R, tal que, $B \neq A_{i_j} \forall j \in \{1, ..., k\}$ e b_1 e $b_2 \in$ dom(B) tal que $b_1 \neq b_2$.

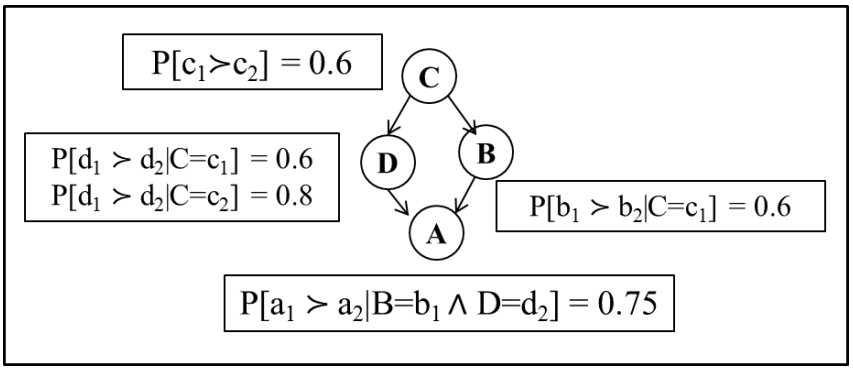


Figura 4.2: Exemplo de Rede de Preferências Bayesianas

Em tarefas de classificação, *Redes Bayesianas* são usadas para classificar tuplas. Já no cenário de Mineração de Preferências as RPBs são usadas para comparar pares de tuplas. Cada probabilidade $P[E_2|E_1]$ nas tabelas da RPB representa uma regra de preferência contextual probabilística (*cp-rule*), onde E_1 é o contexto e E_2 é a preferência. Uma regra de preferência contextual probabilística em um nó X do grafo G, representa o quanto um valor de atributo é preferido a outro valor dependendo dos valores que seus pais assumem. Por exemplo, a probabilidade $P[d_1 \succ d_2|C = c_1] = 0.6$, significa que a probabilidade de $D = d_1$ ser preferível a $D = d_2$ é de 60%, dado o contexto $C = c_1$.

Uma RPB compara duas novas tuplas utilizando uma *ordem parcial estrita*. Para compreender como é feita a comparação de tuplas por essa ordem de preferência, considere o Exemplo 4.1.

Exemplo 4.1. Considere a RPB **PNet** ilustrada na Figura 4.2. Essa **PNet** permite inferir uma ordem de preferência sobre o esquema relacional $R = (A, B, C, D)$, e de acordo com essa ordem suponha que a tupla $u = (a_1, b_1, c_1, d_1)$ é preferível a tupla $v = (a_2, b_2, c_1, d_2)$. Para chegar a essa conclusão, é preciso executar as seguintes etapas:

1. Primeiro, é necessário encontrar o conjunto $\Delta(u, v)$, isto é, o conjunto de atributos em que u e v diferem. Neste exemplo $\Delta(u, v) = \{A, B, D\}$.
2. Num segundo momento, é preciso encontrar o conjunto $min(\Delta(u, v)) \subseteq \Delta$, ou seja, aqueles atributos que não têm pais em Δ de acordo com a **PNet** (Figura 4.2). Neste exemplo $min(\Delta(u, v)) = \{D, B\}$. Sendo assim, para inferir que u é preferível a v é necessário e suficiente que $u[D] \succ v[D]$ e $u[B] \succ v[B]$.
3. A terceira parte consiste em calcular as seguintes probabilidades:
   (a) p_1: Probabilidade de u ser preferível a v.
   Neste exemplo: $u \succ v = P[d_1 \succ d_2 | C = c_1] * P[b_1 \succ b_2 | C = c_1] \Rightarrow 0.6 * 0.6 = 0.36$
   (b) p_2: Probabilidade de v ser preferível a u.
   Neste exemplo: $v \succ u = P[d_2 \succ d_1 | C = c_1] * P[b_2 \succ b_1 | C = c_1] \Rightarrow 0.4 * 0.4 = 0.16$

Para comparar u e v, o método seleciona o maior valor entre as probabilidades p_1 e p_2. Como $p_1 > p_2$, o CPrefMiner infere que u é preferível a v. Caso $p_1 = p_2$, a preferência sobre as duas tuplas é inferida aleatoriamente. A partir dessa comparação e da preferência real das tuplas, é possível avaliar a qualidade do modelo de preferência.

Como avaliar a qualidade de uma RPB no cenário *Crisp*: Considere as seguintes variáveis denotando as cardinalidades dos respectivos conjuntos: (1)M: O BDP-C de teste $\mathcal{P}$; (2) N_r: Conjunto de pares de tuplas em $\mathcal{P}$ ordenadas corretamente pela ordem de preferência da **PNet**; (3) N_a: Conjunto de pares de tuplas em $\mathcal{P}$ ordenadas corretamente pelo módulo aleatório; (4)N_c: Conjunto de pares de tuplas que foram comparadas pela **PNet** (corretas e incorretas). As seguintes medidas são propostas para avaliar a qualidade do modelo de preferência minerado:

1. **Acurácia** (acc): definida por $acc(PNet, \mathcal{P}) = \frac{N_r + N_a}{M}$.
2. **Revocação** ($revoc$): definida por $rev(PNet, \mathcal{P}) = \frac{N_r}{M}$.
3. **Precisão** ($prec$): definida por $prec(PNet, \mathcal{P}) = \frac{N_r}{N_c}$.
4. **Taxa de Comparabilidade** (tc): definida por $tc(PNet, \mathcal{P}) = \frac{N_c}{M}$.

5. **Taxa de Aleatoriedade** (ta): definida por $ta(PNet, \mathcal{P}) = \frac{N_a}{M}$.

O Problema de Mineração de Preferências Contextuais no Cenário *Crisp* : O problema de mineração de preferências contextuais consiste em, dado um banco de dados de preferência *crisp*, retornar uma RPB tendo boa qualidade com respeito às medidas de acurácia, revocação, precisão, taxa de comparabilidade e taxa de aleatoriedade.

Este problema foi introduzido por de Amo *et al.* (2012a) e o algoritmo CPrefMiner foi desenvolvido para solucioná-lo. Por ser a base principal desta pesquisa, há a necessidade de mostrar os detalhes da implementação deste algoritmo na próxima seção deste capítulo.

4.2 O Algoritmo CPrefMiner

O *CPrefMiner* é um método qualitativo para mineração de preferências contextuais *crisp*. Este método recebe como entrada um banco de dados de preferência *crisp* e retorna uma rede de preferência *bayesiana* para ser usada na comparação de novas tuplas.

Basicamente, a tarefa de construção de uma rede de preferência bayesiana consiste em duas etapas: (1) descoberta do grafo G representando a estrutura da rede de preferências e (2) o cálculo das tabelas de probabilidades condicionais de cada nó do grafo G. Para melhor entendimento, cada uma dessas etapas é descrita nas subseções a seguir.

4.2.1 Etapa 1 - Descoberta do grafo G

Esta primeira etapa é feita por um Algoritmo Genético (AG) (Larranaga *et al.*, 1996). A ideia geral do AG é escolher a partir de uma população de indivíduos, o melhor indivíduo de acordo com uma função de *fitness* associada. No trabalho proposto por de Amo *et al.* (2012a) cada indivíduo da população é um grafo direcionado acíclico representando a topologia da RPB sobre um esquema relacional.

O AG proposto evolui uma população de grafos durante certo número de iterações, e retorna o melhor grafo de acordo com uma função *score*. A principal ideia da função *score* é associar um número real no intervalo [-1,1] para cada estrutura candidata G da população de grafos gerados pelo AG. Esse valor estima quão bem o AG captura as dependências entre os atributos de um banco de dados de preferências $\mathcal{P}$. Neste caso, cada aresta (x, y) do grafo G é "punida"ou "recompensada", de acordo com o arco (X, Y) em G e o *grau de dependência* correspondente do par (X, Y) com respeito à $\mathcal{P}$.

O cálculo do *grau de dependência* de um par de atributos (X, Y) com respeito a um banco de dados de preferências $\mathcal{P}$ ilustrado pelo Algoritmo 1, é um número real que estima o quanto os valores do atributo Y são influenciados pelos valores do atributo X. A fim de compreender como é feito o cálculo através deste algoritmo considere um exemplo.

Exemplo 4.2. Suponha um esquema relacional $R = (A, B, C)$, o banco de dados de preferência P e duas estruturas G_1 e G_2 candidatas à RPB ilustrados na Figura 4.3. Para

Algoritmo 1: Grau de Dependência de um Par de Atributos [(de Amo *et al.*, 2012a)]

Entrada: $\mathcal{P}$: um banco de dados de preferência *crisp*; (X, Y): um par de atributos; dois limites $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$.

Saída: O Grau de Dependência de (X, Y) com respeito à $\mathcal{P}$

1 **para** *cada par* $(y, y') \in$ $\boldsymbol{dom}(Y) \times \boldsymbol{dom}(Y)$, $y \neq y'$ *e* (y, y') *comparável* **faça**

2 **para** *cada* $x \in$ $\boldsymbol{dom}(X)$ *onde* x *é a causa para* (y, y') *serem comparáveis* **faça**

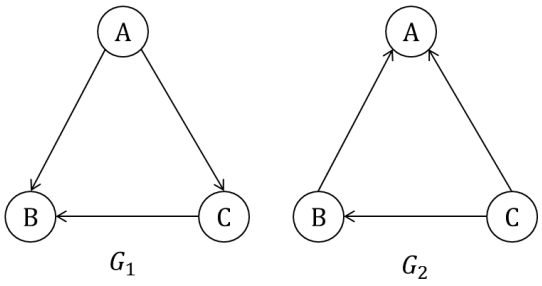
3 Seja $f_1(S_{x|(y,y')}) = \max\{N, 1 - N\}$, onde
$$N = \frac{|\{(t, t') \in S_{x|(y,y')} : t > t' \wedge (t[Y] = y \wedge t'[Y] = y')|\}}{|S_{x|(y,y')}|}$$

4 Seja $f_2(T_{yy'}) = \max\{f_1(S_{x|(y,y')}) : x \in$ $\mathbf{dom}(X)\}$

5 Seja $f_3((X, Y), \mathcal{P}) = \max\{f_2(T_{yy'}) : (y, y') \in$ $\mathbf{dom}(Y) \times \mathbf{dom}(Y)$, $y \neq y'$, (y, y') comparável$\}$ **retorne** $f_3((X, Y), \mathcal{P})$

saber qual estrutura representa melhor uma RPB para este banco de dados é necessário calcular seu valor *score*. O primeiro passo da função *score* é o cálculo do *grau de dependência* de um par de atributos. Para este exemplo, suponha que $\alpha_1 = 0.1$ e $\alpha_2 = 0.2$. Para calcular o *grau de dependência* do par (A, B) com respeito à P é preciso seguir os três passos:

	u				***v***			
Id	**A**	**B**	**C**		**A**	**B**	**C**	
1	a_2	b_1	c_1		a_3	b_3	c_4	$\mathbf{T}_{(b_1,b_3)}$
2	a_1	b_2	c_1		a_4	b_4	c_2	$\mathbf{T}_{(b_2,b_4)}$
3	a_2	b_2	c_2		a_3	b_3	c_3	$\mathbf{T}_{(b_2,b_3)}$
4	a_1	b_2	c_3		a_1	b_3	c_4	
5	a_1	b_3	c_3		a_1	b_2	c_4	
6	a_1	b_3	c_3		a_1	b_2	c_4	
7	a_1	b_3	c_3		a_1	b_2	c_4	
8	a_1	b_3	c_4		a_1	b_2	c_3	
9	a_1	b_3	c_5		a_1	b_2	c_3	
10	a_2	b_3	c_5		a_2	b_2	c_5	
11	a_2	b_5	c_5		a_2	b_6	c_5	$\mathbf{T}_{(b_5,b_6)}$
12	a_2	b_5	c_3		a_2	b_6	c_3	
13	a_2	b_5	c_3		a_2	b_6	c_3	
14	a_2	b_6	c_3		a_2	b_5	c_3	
15	a_3	b_3	c_3		a_3	b_4	c_4	$\mathbf{T}_{(b_3,b_4)}$



G_1 G_2

Figura 4.3: Banco de Dados de Preferência **P** e duas estruturas G_1 e G_2 candidatas à RPB

Passo 1: No primeiro laço do Algoritmo 1, encontre todos os conjuntos $T_{b,b'}$. Calcule o suporte de cada conjunto encontrado e verifique se o par (b, b') é comparável. Neste exemplo:

- $Suporte(T_{b_1,b_3}, P) = \frac{|T_{b_1,b_3}|}{|P|} = \frac{1}{15} = 0.067$.
- $Suporte(T_{b_2,b_4}, P) = \frac{|T_{b_2,b_4}|}{|P|} = \frac{1}{15} = 0.067$.
- $Suporte(T_{b_2,b_3}, P) = \frac{|T_{b_2,b_3}|}{|P|} = \frac{8}{15} = 0.53$.

- $Suporte(T_{b_5,b_6}, P) = \frac{|T_{b_5,b_6}|}{|P|} = \frac{4}{15} = 0.27$.
- $Suporte(T_{b_3,b_4}, P) = \frac{|T_{b_3,b_4}|}{|P|} = \frac{1}{15} = 0.067$.

Um par (b, b') é considerado *comparável*, se o valor do suporte de seu conjunto $T_{b,b'}$ for $\geq \alpha_1$. Portanto, neste exemplo os conjuntos T_{b_1,b_3}, T_{b_2,b_4} e T_{b_3,b_4} são eliminados, pois o suporte de todos eles são inferiores ao valor de α_1. Entrando no segundo laço do Algoritmo 1 segue-se para o segundo passo.

Passo 2: Para cada conjunto $T_{b,b'}$ com suporte maior que α_1, encontre os conjuntos $S_{a|(b,b')}$. Calcule o suporte de todos os conjuntos e verifique quais os valores do atributo A são considerados *causa* para que o par (b, b') seja comparável. O conjunto $S_{a|(b,b')}$ é um subconjunto de $T_{b,b'}$ contendo os pares (b, b'), tal que, $u[A] = v[A] = a$.

- Encontre os conjuntos S do conjunto T_{b_2,b_3}: Para este conjunto T, dois conjuntos S são encontrados, são eles: $S_{a_1|(b_2,b_3)}$ e $S_{a_2|(b_2,b_3)}$. Cujo valores de suporte são:
  - $Suporte(S_{a_1|(b_2,b_3)}, P) = \frac{|S_{a_1|(b_2,b_3)}|}{\cup a' \in dom(A) S_{a'|(b_2,b_3)}} = \frac{6}{7} = 0.857$
  - $Suporte(S_{a_2|(b_2,b_3)}, P) = \frac{|S_{a_2|(b_2,b_3)}|}{\cup a' \in dom(A) S_{a'|(b_2,b_3)}} = \frac{1}{7} = 0.14$

  Um valor a é considerado a *causa* para um par (b, b') ser *comparável*, quando o suporte do seu conjunto $S_{a|b,b'}$ for $\geq \alpha_2$. Logo, neste exemplo, o conjunto $S_{a_2|(b_2,b_3)}$ é eliminado, pois seu suporte é inferior ao valor de α_2.

  Ainda nesta etapa, calcula-se o valor das funções f_1 e f_2. A função f_1 do conjunto $S_{a_1|(b_2,b_3)}$ é dado por: $f1 = max\{N, 1 - N\} = max\{\frac{5}{6}, \frac{1}{6}\} = \frac{5}{6} = 0.83$ e a função $f_2 = max\{f_1\}$. Como neste exemplo só existe um conjunto S com suporte superior a α_2, então o valor da função $f_2 = f_1 = 0.83$.
- Encontre os conjuntos S do conjunto T_{b_5,b_6}: Para este conjunto T só existe um conjunto S definido por: $S_{a_2|(b_5,b_6)}$ cujo valor do suporte é:
  - $Suporte(S_{a_2|(b_5,b_6)}, P) = \frac{|S_{a_2|(b_5,b_6)}|}{\cup a' \in dom(A) S_{a'|(b_5,b_6)}} = \frac{4}{4} = 1.$

  Como o valor do suporte é maior que o valor de α_1, é preciso calcular o valor da função f_1. O valor para a função neste exemplo é $f_1 = max\{N, 1-N\} = max\{\frac{3}{4}, \frac{1}{4}\} = \frac{3}{4} = 0.75$. E consequentemente, o valor de $f_2 = f_1 = 0.75$

Após calcular os valores de f_2 para todos os conjuntos T cujo suporte é maior que α_1, o terceiro passo é executado.

Passo 3: No terceiro passo é calculado o *grau de dependência* do par de atributos (A, B) pela função $f_3 = max\{f_1(T)\}$. Neste exemplo, $f_3 = max\{0.83, 0.75\}$, logo, o *grau de dependência* do par (A, B) com respeito ao banco de dados de preferência $\mathcal{P}$ é $f_3 = 0.83$.

Executando os três passos para todos os pares possíveis do esquema relacional R, os seguintes valores de graus de dependência são obtidos: $(B, A) = 0$, $(A, C) = 0.66$, $(C, A) = 0$, $(B, C) = 0$ e $(C, B) = 1$. Com os valores do grau de dependência de todos os pares de atributos calculados, é possível calcular o valor *score* de estrutura candidata, dado pela função de *fitness*:

$$score(G, P) = \frac{\sum_{X,Y}(g(X, Y), G)}{n(n-1)} \tag{4.1}$$

Onde X e Y são atributos do esquema relacional, n é a quantidade de atributos do esquema relacional. A função g é calculada pelas seguintes regras:

1. Se $f_3((X, Y), P) \geq 0.5$ e a aresta $(X, Y) \in G$, então $g((X, Y), G) = f_3((X, Y), P)$. Essa primeira regra ilustra a situação em que existe uma forte dependência entre (X, Y) e a aresta (X, Y) existe na estrutura candidata, portanto, g é "recompensando"com o valor f_3.
2. Se $f_3((X, Y), P) \geq 0.5$ e a aresta $(X, Y) \notin G$, então $g((X, Y), G) = -f_3((X, Y), P)$. A segunda regra ilustra a situação que existe uma forte dependência entre os atributos (X, Y) e a aresta (X, Y) não existe na estrutura candidata, então g é "punido"com o valor f_3.
3. Se $f_3((X, Y), P) < 0.5$ e a aresta $(X, Y) \notin G$, então $g((X, Y), G) = 1$. A terceira regra ilustra a situação que existe uma fraca dependência entre os atributos e aresta realmente não existe na estrutura candidata, portanto, g é "recompensado"com o valor 1.
4. Se $f_3((X, Y), P) < 0.5$ e a aresta $(X, Y) \in G$, então $g((X, Y), G) = 0$. Finalmente a quarta regra, ilustra a situação que existe uma fraca dependência entre os atributos e a estrutura candidata contém a aresta, logo, g é "punido"com o valor 0.

Com essas regras é possível calcular o *score* das duas estruturas candidatas ilustradas na Figura 4.3. Os valores *scores* para as estruturas candidatas G_1 e G_2 são 0.915 e 0.085, respectivamente. Portanto, para este exemplo, a estrutura candidata G_1 seria escolhida, pois, possui maior valor *score*.

Nesta etapa, é calculada a função *score* para todas as estruturas candidatas, de acordo com a população do algoritmo genético. Ao final da execução do AG, a rede com melhor valor *score* é retornada para realizar o cálculo das tabelas de probabilidades condicionais de cada nó da rede.

4.2.2 Etapa 2 - Cálculo das Tabelas de Probabilidades Condicionais

A segunda fase da construção da RPB, consiste em calcular as tabelas de probabilidades condicionais de cada nó do grafo G que representa a estrutura da RPB. Nesta etapa é usado o Princípio de Probabilidade Máxima (Jansen e Nielsen, 2007), que se baseia na frequência em que cada valor de atributo aparece no banco de dados para estimar uma probabilidade. Para entender melhor como funciona este cálculo, considere o seguinte exemplo:

Exemplo 4.3. Considere a estrutura candidata G_1 e o banco de dados de preferência do exemplo anterior, ilustrado pela Figura 4.3. Suponha o cálculo da probabilidade $P[c_3 \succ c_4|A = a_1]$. Para estimar essa probabilidade, é necessário calcular $\frac{N(C=c_3,A=a_1)}{N(C=c_3,A=a_1)+N(C=c_4,A=a_1)}$, onde $N(C = c_3, A = a_1)$ é o número de casos em que $N(C = c_3, A = a_1)$ é preferível a $N(C = c_4, A = a_1)$.

Observando a instância do banco de dados ilustrado na Figura 4.3, verifica-se que os pares de tuplas que $(c_3 \succ c_4|a_1)$ são aqueles com $Ids = \{4, 5, 6, 7\}$, isto é, existem 4 casos em que c_3 é preferível a c_4 quando o contexto é $(A = a_1)$. Já o caso $(c_4 \succ c_3|a_1)$ aparece apenas em um par de tuplas, cujo $Id = \{8\}$. Logo, estima-se a probabilidade $P[c_3 \succ c_4|A = a_1] = \frac{4}{(4+1)} = 0.8$ e consequentemente a probabilidade $P[c_4 \succ c_3|A = a_1] = \frac{1}{(4+1)} = 0.2$ é o valor complementar no intervalo [0,1].

Este calculo é feito para todos os valores possíveis de um banco de dados de preferência. Portanto, uma tabela de probabilidade condicional da RPB, é um conjunto finito de probabilidades condicionais da forma $P[E_1|E_2]$, onde o evento E_1 da condição é o contexto e o evento E_2 é a preferência. Neste exemplo, a probabilidade do valor $C = c_3$ ser preferível a $C = c_4$ dado o contexto $A = a_1$ é de 80%.

Uma vez construído o modelo de preferência, o mesmo é utilizado para descobrir a preferência sobre novos pares de tuplas e sua qualidade é avaliada de acordo com as medidas de avaliações definidas na Seção 4.1.

4.3 Considerações do Capítulo

Neste Capítulo foi mostrado a formalização do problema de mineração de preferências no cenário *Crisp*, bem como a apresentação em detalhes do algoritmo de mineração de preferências CPrefMiner (de Amo *et al.*, 2012a), (de Amo *et al.*, 2013) para solucionar o problema em questão. A apresentação em detalhes deste algoritmo se faz necessária, uma vez que, o algoritmo proposto nesta dissertação é baseado no CPrefMiner. O próximo capítulo mostra detalhes da proposta do algoritmo*FuzzyPrefMiner*, desenvolvido para trabalhar com representação de preferências *fuzzy*.

Capítulo

5

Mineração de Preferências Contextuais Fuzzy

Este capítulo tem como objetivos: (1) formalizar o problema de mineração de preferências abordado nesta dissertação e (2) apresentar o algoritmo proposto para solucionar o problema em questão.

Sendo assim, a Seção 5.1 aborda os conceitos necessários para a formalização do problema de mineração de preferências *fuzzy* e seguidamente, a Seção 5.2 descreve o algoritmo *FuzzyPrefMiner* proposto e implementado para solucionar o problema de mineração de preferências contextuais no cenário *fuzzy*.

5.1 Formalização do Problema

O problema geral de mineração de preferências *fuzzy* tratado nesta dissertação ilustrado pela Figura 5.1 consiste basicamente em dividir um BDP-F $\mathcal{P}$ em vários bancos de dados menores de acordo com uma faixa de preferência. O algoritmo FuzzyPrefMiner é aplicado em cada BDP-F $\mathcal{P}'$ para extrair uma RBP, ou seja, de um conjunto de bancos de dados de preferências *fuzzy* é extraído um conjunto de redes de preferências *bayesianas*. Além disso, a divisão do BDP-F $\mathcal{P}$ em vários bancos de dados menores auxilia no processo de treinamento de um classificador a partir da Ferramenta *Weka* [1]. Portanto, o modelo de preferência é composto por um conjunto de redes *bayesianas* de preferências e um classificador $\mathcal{M}$ treinado. Este modelo de preferências pode ser usado para inferir preferências tanto no cenário *crisp* quanto no cenário *fuzzy* como ilustrado na Figura 5.1.

[1] http://www.cs.waikato.ac.nz/ml/index.html

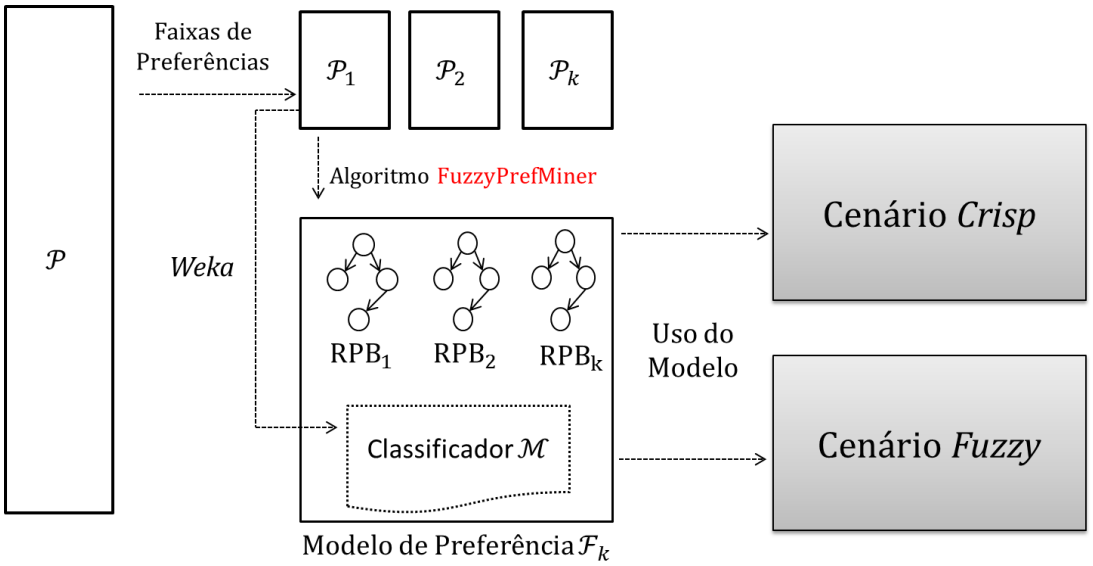


Figura 5.1: O Problema Geral de Mineração de Preferências Contextuais *Fuzzy*

A Figura 5.2 exemplifica o processamento dos dados pela divisão de faixas de preferências, a obtenção do modelo de preferências e as respostas do modelo de preferência quando aplicados tanto no cenário *crisp*, quanto no cenário *fuzzy*. O BDP-F $\mathcal{P}$ (Figura 5.1) é um banco de dados composto por um conjunto de *triplas* (como exemplificado na Figura 5.2c). Este BDP-F corresponde a um subconjunto de informações contidas em uma matriz de preferência como a ilustrada pela Figura 5.2(b), que por sua vez, é extraída de um conjunto de tuplas avaliadas pelo usuário por uma função *score* (Figura 5.2(a)). O modelo de preferência *fuzzy* tem o objetivo de predizer, dado duas novas tuplas u e v, qual a tupla preferida. Caso o modelo seja usado no cenário *Crisp*, o mesmo retorna uma resposta do tipo "sim"ou "não", ou a tupla u é preferível a v, ou v é preferível a u. Já, se o modelo for usado no cenário *fuzzy*, o mesmo retorna o *grau de preferência* da tupla u com respeito a tupla v.

Para extrair o modelo de preferência a partir de um BDP-F, o mesmo é dividido em vários bancos de dados de preferência *fuzzy* menores. Essa divisão é feita de acordo com as faixas de preferências definidas por:

Definição 5.1. (Faixas de Preferências) Seja $\mathcal{P}$ um banco de dados de preferência *fuzzy* e $I = [gp_{min}, gp_{max}]$ o intervalo de graus de preferências de $\mathcal{P}$, isto é, gp_{min} é o menor grau de preferência em $\mathcal{P}$ e gp_{max} o maior grau de preferência em $\mathcal{P}$, tal que, $gp > 0.5$. Uma faixa de preferência sobre $\mathcal{P}$ é um par $(k, \{I_1, \ldots, I_k\})$, onde k é um valor inteiro que indica em quantas partes o banco $\mathcal{P}$ é dividido e $\{I_1, ..., I_k\}$ representa o intervalo de graus de preferências de cada parte de $\mathcal{P}$.

O topo da Figura 5.2(d) ilustra o particionamento do BDP-F (Figura 5.2(c)) em duas faixas de preferências, tal que, $\mathcal{P}_1$ contém *triplas* no intervalo $I_1 = (0.64, 0.8]$ e $\mathcal{P}_2$ contém *triplas* no intervalo $I_2 = (0.8, 0.94]$. Ao particionar o BDP-F em k faixas de preferências, as *triplas* são colocadas ordenadas, isto é, todas com $gp > 0.5$. Isso se

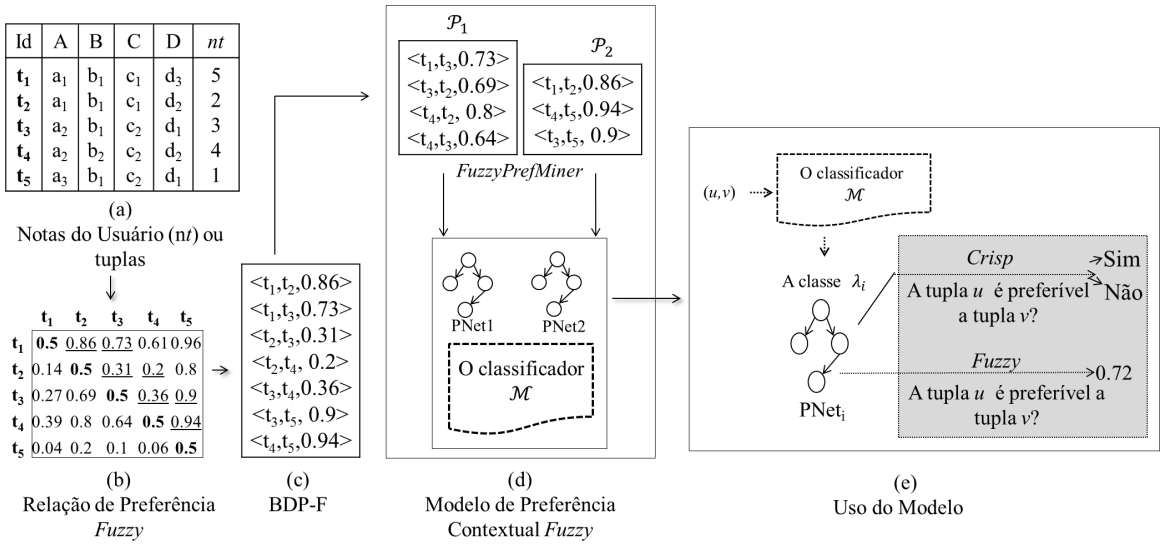


Figura 5.2: Processo da Mineração de Preferências Contextuais no cenário *Fuzzy*

torna necessário, pois no momento da extração da rede de preferências é preciso treinar a ordem de preferência entre as tuplas que compõem o BDP-F. Desse conjunto de k bancos de dados de preferência *fuzzy* dividido pelas faixas de preferências, é extraído um conjunto $S = \{PNet_1, \ldots, PNet_k\}$ de redes de preferências *bayesianas*.

Além disso, a divisão do BDP-F em faixas de preferências é usada para o treinamento do classificador $\mathcal{M}$. As faixas de preferências podem ser vistas como classes e essas classes podem ser nomeadas como λ_i, por exemplo, para um intervalo $I_i = (a_i, b_i]$ a classe $\lambda_i = \frac{a_i+b_i}{2}$ é considerada. Dessa forma, as *triplas* (u, v, n) em $\mathcal{P}_i$ são caracterizadas por tuplas u e v que tem uma média de "intensidade"de preferência λ_i entre elas.

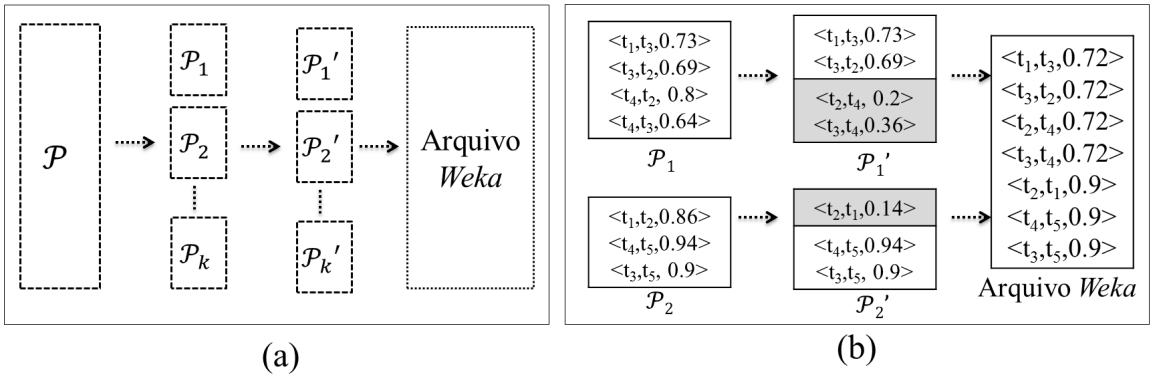


Figura 5.3: (a) Diagrama Geral da Transformação por Faixas de Preferências e (b) Exemplo da Transformação

A ideia geral de como transformar um BDP-F $\mathcal{P}_i$ em um arquivo no formato padrão considerado pela ferramenta *Weka* é ilustrada pela Figura 5.3(a). Ou seja, o BDP-F é dividido em faixas de preferências e a partir dessas faixas é possível compor um arquivo Weka para treinamento do classificador. Por exemplo, o BDP-F ilustrado na Figura 5.2(c) foi dividido em duas faixas de preferências gerando dois subconjuntos $\mathcal{P}_1$ e $\mathcal{P}_2$ (Figura

5.3(b)). No arquivo *Weka*, as triplas de cada BDP-F dividido pela faixas de preferências são classificados pelo valor médio (λ_i) do intervalo I_k pertencente à faixa de preferência k.

Ao gerar o arquivo *Weka* os dados não ficam na ordem de preferência ($gp > 0.5$) (como nas partições para treinar a RPB), pois o objetivo do classificador é apenas identificar a *intensidade de preferência*, e não a ordem entre as tuplas. Portanto, o treinamento do classificador é realizado com aproximadamente 50% das *triplas* com $gp > 0.5$ e as outras 50% de *triplas* com $gp < 0.5$ classificados pela mesma intensidade de preferência (λ_i), ou seja, o valor médio do intervalo estipulado pela faixa de preferência k.

O Modelo de Preferência Proposto: O modelo de preferência *fuzzy* proposto nesta dissertação é uma estrutura $\mathcal{F}_k = <\mathcal{M}, PNet_1, ..., PNet_k>$, onde $\mathcal{M}$ é um modelo de classificação extraído das faixas de preferências $\{\mathcal{P}'_1, ..., \mathcal{P}'_k\}$ (considerando elementos em $\mathcal{P}_i$ como (u, v, n, λ_i) com uma dimensão extra, ou seja, a "*classe*" λ_i), e $PNet_i$ é uma RPB extraída de uma faixa de preferência $\mathcal{P}_i$, para todo $i = 1, ..., k$.

É importante enfatizar que a RPB extraída a partir de um banco de dados de preferência *fuzzy* (denotada por RPB-f) não é a mesma RPB extraída de um banco de dados de preferência *crisp* (denotada por RPB-c). Pois uma RPB-f é extraída de um conjunto de dados que possui mais informações.

Como a estrutura $\mathcal{F}_k$ compara duas tuplas: Seja (u, v) um novo par de tuplas. O primeiro passo para comparar as duas tuplas é inferir a intensidade de preferência que existe entre elas. O classificador $\mathcal{M}$ é responsável por essa tarefa. Ele é executado sobre o par (u, v) e retorna a classe λ_i para este par. Então, a RPB-f $PNet_i$ extraída da faixa de preferência I_i é executada sobre o par (u, v) para decidir a ordem de preferência. Essa execução segue os mesmos passos mostrados no Exemplo 4.1 do Capítulo 4. Se a $PNet_i$ decide que u é preferível a v, a predição final é: u é preferível a v com grau de preferência λ_i. Caso contrário, a predição final é: u é preferível a v com grau de preferência $1 - \lambda_i$.

É importante ressaltar que o modelo de preferência $\mathcal{F}_k$ pode ser usado para comparar duas tuplas tanto no cenário *Crisp* quanto no cenário *Fuzzy* como ilustra a Figura 5.2(e).

Como avaliar a qualidade do modelo de preferência *Fuzzy*: No cenário *Crisp* a avaliação da qualidade do modelo acontece como no Algoritmo CPrefMiner. Já no cenário *Fuzzy*, avaliar a qualidade do modelo é um processo envolvendo comparação de graus de preferências. Portanto, uma vez extraído o grau de preferência (gp_p) pelo modelo $\mathcal{F}_k$ e conhecendo o grau de preferência (gp_r) fornecido pelo usuário, um parâmetro $\kappa > 0$ é considerado para verificar quão boa foi a predição do modelo de preferência. O gp_p é considerado correto se duas condições são verificadas: (1) $|gp_r - gp_p| \leq \kappa$ e (2) ($gp_r \geq 0.5$ e $gp_p \geq 0.5$) ou ($gp_r < 0.5$ e $gp_p < 0.5$). Para ilustrar este processo, considere um exemplo:

Exemplo 5.1. Considere as tuplas u e v do Exemplo 4.1. Suponha que a entrada é o banco de dados de preferência *fuzzy* ilustrado na Figura 5.2(d), particionado em duas faixas de preferências $\mathcal{P}_1$ e $\mathcal{P}_2$. As classes correspondentes são $\lambda_1 = 0.72$ (gp médio do intervalo (0.64,0.8]) e $\lambda_2 = 0.9$ (gp médio do intervalo (0.8,0.94]). Suponha que o grau de preferência real fornecido pelo usuário é $gp_r(u, v) = 0.78$ e que o classificador $\mathcal{M}$ classifique o par (u, v) com a classe λ_1. O classificador infere apenas a intensidade de preferência entre as duas tuplas, ele não prediz a ordem de preferência entre elas. Essa tarefa é de responsabilidade da $PNet_1$ extraída da partição $\mathcal{P}_1$.

Para fins de ilustração, suponha que a $PNet_1$ seja a mesma ilustrada na Figura 4.2. Ela compara as duas tuplas u e v como descrito no Exemplo 4.1, calculando as probabilidades $p_1 = 0.36$ e $p_2 = 0.16$. Como $p_1 > p_2$ então o grau de preferência predito é $gp_p = \lambda_1 = 0.72$. Ambos os graus de preferências (gp_r e gp_p) são ≥ 0.5, ou seja, verifica a segunda propriedade. Considere que o parâmetro $\kappa = 0.05$ e analise se a primeira propriedade é garantida. Como $|0.78 - 0.72| = 0.06 > 0.05$, então, se conclui que o modelo de preferência *fuzzy* errou a predição do grau de preferência da tupla u com respeito a tupla v.

O Problema de Mineração de Preferências Contextuais *Fuzzy*: Dado um esquema relacional R, um BDP-F $\mathcal{P}$, uma faixa de preferência $(k, \{I_1, ..., I_k\})$, retornar um modelo de preferência *fuzzy* $\mathcal{F}_k$ tendo boa qualidade com respeito a acurácia, revocação, precisão, taxa de comparabilidade e taxa de aleatoriedade, dado um parâmetro κ.

Com o problema de mineração de preferência *fuzzy* formalizado, a próxima Seção descreve o algoritmo proposto para solucionar este problema.

5.2 O Algoritmo *FuzzyPrefMiner*

O Algoritmo *FuzzyPrefMiner* proposto nesta dissertação para extrair as redes de preferências *bayesianas* a partir de um banco de dados de preferência *fuzzy*, envolve a modificação do núcleo do algoritmo CPrefMiner (de Amo *et al.*, 2012a) descrito no Capítulo 4.

Essa modificação foi realizada tanto na aprendizagem da topologia da rede de preferência, quanto no cálculo das tabelas de probabilidades condicionais. O processo geral do *FuzzyPrefMiner* é mostrado no Algoritmo 2.

Nesta seção, são apresentados os detalhes das partes do método que foi modificada para trabalhar com representação de preferência *fuzzy*. As modificações foram realizadas na etapa de Aprendizagem da Topologia da Rede representada no Algoritmo 2 pelo procedimento *"Extração"* e no Cálculo das Tabelas de Probabilidades Condicionais representado pelo procedimento *"ComputeTabela"*.

Algoritmo 2: O Algoritmo FuzzyPrefMiner

Entrada: Um BDP-F $\mathcal{P}_i$ sobre um esquema relacional $R(A_1, ..., A_n)$ *correspondendo a uma faixa de preferência em* $\mathcal{P}$;

Saída: Uma RPB-f **PNet**

1 $\boxed{Extração(\mathcal{P},\ G)}$;
2 **para** *cada* $i = 1, \ldots, n$ **faça**
3 $Pais(G, A_i) = [A_{i_1}, ..., A_{i_m}]$;
4 $\boxed{\textbf{\textit{ComputeTabela}}(A_i,\ \textbf{\textit{Pais}}(A_i, G)) = [CProb_1, ..., CProb_l]}$;
5 **retorne** *PNet*

5.2.1 Aprendizagem da Topologia da Rede

O procedimento *Extração*$(\mathcal{P}, G)$ (veja Algoritmo 3) é responsável pela aprendizagem da topologia G a partir de um banco de dados de preferência *fuzzy* $\mathcal{P}$. G é um grafo cujo vértices são atributos do esquema relacional $R = (A_1, ..., A_n)$ e uma aresta (A_i, A_j) em G, significa que a preferência nos valores do atributo A_j dependem dos valores que o atributo A_i assume. Isto é, A_i é o contexto para as preferências sobre o atributo A_j.

Algoritmo 3: Procedimento *Extração* (de Amo *et al.*, 2012a)

Entrada: $\mathcal{P}$: um BDP-F sobre um esquema relacional $R(A_1, ..., A_n)$; β: tamanho da população; ϑ: número de gerações; γ: número de ordenações dos atributos.

Saída: Um grafo $G = (V, E)$, com vértices $V \subseteq \{A_1, ..., A_n\}$ sendo o melhor grafo de acordo com a função de *fitness*.

1 **para** *cada* $i = 1, \ldots, \gamma$ **faça**
2 Aleatoriamente gere uma ordenação de atributos;
3 Gere uma população inicial I_0 de indivíduos aleatórios;
4 $\boxed{\textbf{\textit{Avalie os indivíduos em}}\ I_0, \textbf{\textit{ isto é, calcule seu fitness}}}$;
5 **para** *cada* j*-th geração,* $j = 1, \ldots, \vartheta$ **faça**
6 Selecione $\beta/2$ pares de indivíduos em I_j;
7 Aplique o operador *crossover* em cada par, gere uma população de filhos I'_j;
8 Avalie os indivíduos de I'_j;
9 Aplique o operador de mutação sobre I'_j, e avalie os indivíduos mutados;
10 De $I_j \cup I'_j$, selecione o β indivíduos com melhor valor de *fitness*;
11 Pegue o melhor indivíduo após a última geração;
12 **retorne** *O melhor indivíduo de todas* γ *execuções do AG*

Para construir o grafo G, o Algoritmo Genético (Larranaga *et al.*, 1996) gera uma população inicial $\mathbf{P}$ de grafos com vértices em $\{A_1, ..., A_n\}$ e cada grafo $G \in \mathbf{P}$ é avaliado por uma função de *fitness*. O núcleo dessa função de *fitness* também foi modificado para se trabalhar com representação de preferência *fuzzy*.

A principal ideia da função de *fitness* é associar um número real (um *score*) no intervalo [-1,1] para uma estrutura candidata G, que significa quão bem o AG captura as

dependências entre os atributos em um banco de dados de preferência *fuzzy* $\mathcal{P}$. Dessa forma, um arco da rede é "punido"ou "recompensado", de acordo com o arco (X, Y) em G e o *grau de dependência* do arco (X, Y) com respeito a $\mathcal{P}$.

O Novo Grau de Dependência de um Par de Atributos: O *grau de dependência* de um par de atributos (X, Y) com respeito a um BDP-F $\mathcal{P}$, é um número real que estima o quanto as preferências nos valores do atributo Y são influenciados pelos valores do atributo X. Neste novo algoritmo, os graus de preferências das *triplas* são inseridos no cálculo do grau de dependência. Este cálculo é descrito no Algoritmo 4 e para facilitar seu entendimento considere algumas noções:

1. Para cada $y, y' \in \text{dom}(Y)$, $y \neq y'$ denota-se por $T_{y,y'}$ o subconjunto de pares $(t, t') \in \mathcal{P}$, tal que, $t[Y] = y \wedge t'[Y] = y'$ ou $t[Y] = y' \wedge t'[Y] = y$;
2. Se S é um conjunto de *triplas* em $\mathcal{P}$, denota-se por $GP(S)$ o multiconjunto constituído dos graus de preferências das triplas de S (repetições são consideradas como elementos diferentes)
3. Define-se $suporte((T_{y,y'}), \mathcal{P}) = \frac{\sum gp \in GP(T_{y,y'})}{\sum gp \in GP(\mathcal{P})}$.

   O par $(y, y') \in \text{dom}(Y) \times \text{dom}(Y)$ é comparável se $suporte((T_{y,y'}), \mathcal{P})) \geq \alpha_1$, para um limite α_1, $0 \leq \alpha_1 \geq 1$.
4. Para cada $x \in \text{dom}(X)$, denota-se por $S_{x|(y,y')}$ o subconjunto de $T_{y,y'}$ contendo as *triplas* (t, t', n), tal que, $t[X] = t'[X] = x$.
5. Define-se $suporte((S_{x|y,y'}), \mathcal{P}) = \frac{\sum_{gp \in GP(S_{x|(y,y')})} gp}{\sum_{z \in \mathbf{dom}(X), gp \in GP(S_{z|(y,y')})} gp}$.

   O valor x é a *causa* para (y, y') ser *comparável* se $suporte((S_{x|y,y'}), \mathcal{P}) \geq \alpha_2$, para um limite α_2, $0 \leq \alpha_2 \geq 1$.

Algoritmo 4: O Novo Grau de Dependência de um Par de Atributos

Entrada: $\mathcal{P}$: um banco de dados de preferência *fuzzy*; (X, Y): um par de atributos; dois limites $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$.

Saída: O Grau de Dependência de (X, Y) com respeito a $\mathcal{P}$

para *cada par* $(y, y') \in \boldsymbol{dom}(Y) \times \boldsymbol{dom}(Y)$, $y \neq y'$ *e* (y, y') *comparável* **faça**
  para *cada* $x \in \boldsymbol{dom}(X)$ *onde* x *é a causa para* (y, y') *serem comparáveis* **faça**
    Seja $f_1(S_{x|(y,y')}) = \max\{N, 1 - N\}$, onde
    $$N = \frac{\{\sum gp \in GP(S_{x|(y,y'))} : t > t' \wedge (t[Y] = y \wedge t'[Y] = y')\}}{\sum gp \in GP(S_{x|(y,y')})}$$
  Seja $f_2(T_{yy'}) = \max\{f_1(S_{x|(y,y')}) : x \in \mathbf{dom}(X)\}$
Seja $f_3((X, Y), \mathcal{P}) = \max\{f_2(T_{yy'}) : (y, y') \in \mathbf{dom}(Y) \times \mathbf{dom}(Y)$, $y \neq y'$, (y, y') comparável$\}$ **retorne** $f_3((X, Y), \mathcal{T})$

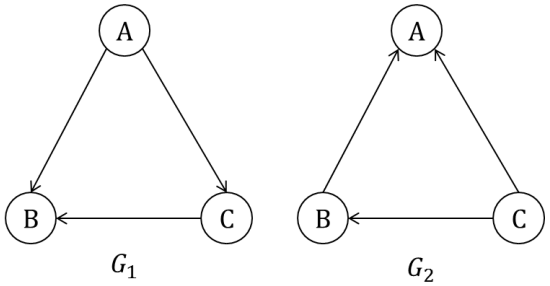
Modificado o núcleo do algoritmo que calcula o *grau de dependência* é necessário realizar o cálculo da função de *fitness* dado pela Equação 5.1.

$$score(G) = \frac{\sum_{X,Y} g((X, Y), G)}{n(n-1)} \tag{5.1}$$

O valor da função g é definida de acordo com as seguintes regras: **(1)** Se $f_3((X, Y), P) \geq 0.5$ e a aresta $(X, Y) \in G$, então $g((X, Y), G) = f_3((X, Y), P)$; **(2)** Se $f_3((X, Y), P) \geq 0.5$ e a aresta $(X, Y) \notin G$, então $g((X, Y), G) = -f_3((X, Y), P)$; **(3)** Se $f_3((X, Y), P) < 0.5$ e a aresta $(X, Y) \notin G$, então $g((X, Y), G) = 1$ e **(4)** Se $f_3((X, Y), P) < 0.5$ e a aresta $(X, Y) \in G$, então $g((X, Y), G) = 0$, como mostrado no Capítulo 4.

Para ilustrar o cálculo da função *score* no cenário *fuzzy* considere o mesmo BDP-F $\mathcal{P}$ do Exemplo 4.2, agora com uma dimensão extra representando o grau de preferência do par de tuplas como ilustrado na Figura 5.4.

	u			*v*				
Id	**A**	**B**	**C**	**A**	**B**	**C**	**gp**	
1	a_2	b_1	c_1	a_3	b_3	c_4	0.67	$T_{(b1,b3)}$
2	a_1	b_2	c_1	a_4	b_4	c_2	0.73	$T_{(b2,b4)}$
3	a_2	b_2	c_2	a_3	b_3	c_3	0.84	
4	a_1	b_2	c_3	a_1	b_3	c_4	0.91	
5	a_1	b_3	c_3	a_1	b_2	c_4	0.72	
6	a_1	b_3	c_3	a_1	b_2	c_4	0.58	$T_{(b2,b3)}$
7	a_1	b_3	c_3	a_1	b_2	c_4	0.94	
8	a_1	b_3	c_4	a_1	b_2	c_3	0.83	
9	a_1	b_3	c_5	a_1	b_2	c_3	0.65	
10	a_2	b_3	c_5	a_2	b_2	c_5	0.96	
11	a_2	b_5	c_5	a_2	b_6	c_5	0.88	
12	a_2	b_5	c_3	a_2	b_6	c_3	0.76	$T_{(b5,b6)}$
13	a_2	b_5	c_3	a_2	b_6	c_3	0.64	
14	a_2	b_6	c_3	a_2	b_5	c_3	0.61	
15	a_3	b_3	c_3	a_3	b_4	c_4	0.83	$T_{(b3,b4)}$



G_1 G_2

Figura 5.4: Banco de Dados de Preferência *Fuzzy* $\mathcal{P}$ e duas estruturas candidatas G_1 e G_2

Exemplo 5.2. Considere o mesmo par de atributos (A,B) do Exemplo 4.2 e BDP-F $\mathcal{P}$ ilustrado na Fig. 5.4. Considere $\alpha_1 = 0.1$ e $\alpha_2 = 0.2$. Seja $T_1 = T(b_1, b_3)$, $T_2 = T(b_2, b_4)$, $T_3 = T(b_2, b_3)$, $T_4 = T(b_5, b_6)$ e $T_5 = T(b_3, b_4)$, o valor dos suportes desses conjuntos são:

- $Suporte(T_{b_1,b_3}, \mathcal{P}) = \frac{0.67}{11.55} = 0.058$.
- $Suporte(T_{b_2,b_4}, \mathcal{P}) = \frac{0.73}{11.55} = 0.063$.
- $Suporte(T_{b_2,b_3}, \mathcal{P}) = \frac{0.84+0.91+0.72+0.58+0.94+0.83+0.65+0.96}{11.55} = 0.56$.
- $Suporte(T_{b_5,b_6}, \mathcal{P}) = \frac{0.88+0.76+0.64+0.61}{11.55} = 0.25$.
- $Suporte(T_{b_3,b_4}, \mathcal{P}) = \frac{0.83}{11.55} = 0.07$.

Portanto T_1, T_2 e T_5 são descartados, pois seus respectivos suportes são menores que α_1. Entrando no segundo *loop* para T_3, dois conjuntos S são encontrados, são eles: $S_{(a_1|b_2,b_3)}$ e $S_{(a_2|b_2,b_3)}$ cujo valores de suporte são:

- $Suporte(S_{a_1|(b_2,b_3)}, \mathcal{P})$=$\frac{4.63}{5.59} = 0.82$
- $Suporte(S_{a_2|(b_2,b_3)}, \mathcal{P})$=$\frac{0.96}{5.59} = 0.17$

Logo $S_{(a_2|b_2,b_3)}$ é descartado, pois o valor é inferior a α_2. Portanto, para $S_{(a_1|b_2,b_3)}$ tem-se $N = \frac{0.91}{4.63} = 0.1965$, $f_1(S_{(a_1|b_2,b_3)}) = max\{N, 1-N\} = 0.8035$ e $f_2(T_3) = 0.8035$. Realizando o mesmo processo para T_4 o valor $f_2(T_4) = 0.7889$ é obtido, portanto, o grau de dependência do par (A,B) é dado por $f_3((A,B),\mathcal{P}) = \max\{0.8035, 0.7889\} = 0.8035$. Os graus de dependências dos demais pares de atributos são $f_3(B,A) = 0$, $f_3(A,C) = 0.79$, $f_3(C,A) = 0$, $f_3(B,C) = 0$ e $f_3(C,B) = 1$.

O *score* da estrutura candidata G_1 ilustrada na Figura 5.4 é dado por $score(G_1,\mathcal{P}) = \frac{(0.8035+1+0.79+1+1+1)}{6} = 0.93$. Enquanto o *score* da segunda estrutura G_2 é $score(G_2,\mathcal{P}) = \frac{(-0.8035+0-0.79+0+1+1)}{6} = 0.067$. Analisando os valores obtidos, G_1 captura melhor as dependências entre os pares de atributos do que G_2.

Os valores encontrados pelo cálculo do grau de dependência no cenário *fuzzy*, dependem dos valores dos graus de preferências dos pares de tuplas pertencentes ao banco de dados de preferência *fuzzy*, ao contrário do grau de dependência calculado no cenário *crisp*, que contabiliza a quantidade de pares de tuplas. A próxima subseção mostra detalhes a modificação realizada na etapa que calcula as tabelas de probabilidades condicionais.

5.2.2 Cálculo das Tabelas de Probabilidades Condicionais

Uma vez encontrado o melhor grafo G que representa a estrutura de uma RPB-f, é preciso calcular as tabelas de probabilidades condicionais de cada vértice do grafo. O cálculo das tabelas de probabilidades condicionais representado pelo procedimento *"ComputeTabela"* do Algoritmo 2 também foi modificado para trabalhar com representação *fuzzy*.

Para estimar as probabilidades condicionais é usado o Princípio de Probabilidade Máxima (Jansen e Nielsen, 2007) assim como no CPrefMiner, só que agora o grau de preferência da *tripla* é incorporado neste cálculo. A intuição por trás deste princípio usa a frequência com que os valores dos atributos aparecem no BDP-F como estimação da probabilidade. Para inserir a informação de grau de preferência neste cálculo, o mesmo foi modificado, de forma que não se limite apenas em usar a frequência com que os valores aparecem no BDP-F, mas a soma dos seus graus de preferências. Para ilustrar este novo procedimento, considere o Exemplo 5.3.

Exemplo 5.3. Considere a mesma estrutura G_1 e banco de dados de preferência *fuzzy* $\mathcal{P}$ do exemplo anterior (Figura 5.4). Suponha o cálculo da probabilidade condicional

$P[c_3 \succ c_4|A = a_1]$. Para estimar essa probabilidade os graus de preferências das *triplas* em que $(c_3 \succ c_4|a_1)$ são considerados. Este cálculo é feito a partir da Equação 5.2.

$$\frac{\sum gp \in N(c_3, c_4|a_1)}{\sum gp \in N(c_3, c_4|a_1) + \sum gp \in N(c_4, c_3|a_1)} \tag{5.2}$$

Neste exemplo, é preciso encontrar as *triplas* em que $(c_3 \succ c_4|a_1)$ e realizar a soma dos seus graus de preferências. Olhando a Figura 5.4, nota-se que as *triplas* em que ocorre essa preferência são aquelas com $Ids = \{4, 5, 6, 7\}$. Já a preferência $(c_4 \succ c_3|a_1)$ só ocorre em uma *tripla* cujo $Id = \{8\}$.

A partir da soma de graus de preferências, a probabilidade condicional é calculada como acontece no CPrefMiner, isto é, $P[c_3 \succ c_4|A = a_1] = \frac{(0.91+0.72+0.58+0.94)}{((0.91+0.72+0.58+0.94)+(0.83))} = 0.79$ e consequentemente a probabilidade $P[c_4 \succ c_3|A = a_1] = 0.21$.

Com essas modificações na etapa de criação do modelo de preferência, o primeiro objetivo é verificar se a inserção do grau de preferência tem influência na qualidade do modelo de preferência. Além disso, quando aplicado no cenário *fuzzy*, o modelo de preferência infere o *grau de preferência* de um par de tuplas. Com essa inferência, é possível retornar de um BDP-F para uma relação de preferência *fuzzy*. Portanto, o segundo objetivo é analisar se as relações de preferências *fuzzy* inferidas pelo modelo de preferência proposto atendem propriedades de consistência das relações de preferências *fuzzy*.

5.3 Considerações do Capítulo

Este capítulo apresenta a primeira contribuição desta pesquisa, isto é, o algoritmo *FuzzyPrefMiner* proposto para solucionar o problema de mineração de preferências contextuais no cenário *Fuzzy*. Num primeiro momento, o problema de mineração de preferências foi formalizado. Seguidamente, foi apresentado as alterações feitas no algoritmo CPrefMiner para obter o *FuzzyPrefMiner*, que trabalha com representação de preferências *fuzzy* na etapa de criação das redes de preferências.

É importante enfatizar que uma RPB-f aplicada no cenário *fuzzy*, pode não garantir propriedades de consistência de relações de preferências *fuzzy*. Portanto, o próximo capítulo desta dissertação apresenta os principais conceitos relacionados a consistência de relações de preferências *fuzzy*, bem como, os métodos de reparação de inconsistência dessas relações.

Capítulo 6

Métodos de Reparação de Inconsistência

Este capítulo tem como principais objetivos: (1) detalhar um método de reparação de inconsistência existente na literatura, pois o mesmo apresenta um algoritmo para avaliar a inconsistência de relações de preferências *fuzzy* utilizado neste tranbalho e (2) apresentar dois novos métodos de reparação de inconsistência de relações de preferência *fuzzy*.

A Seção 6.1 é voltada para apresentar o método BTF (*Baseado em Transitividade Fraca*) (Xu *et al.*, 2013) para reparação de relações de preferências *fuzzy* inconsistentes. Este método é o *baseline* de comparação com os métodos propostos nesta dissertação. Já a Seção 6.2 apresenta a proposta de dois novos métodos de reparação de inconsistência de relações de preferências *fuzzy* baseados na técnica *Range Voting* (de Amo *et al.*, 2014), originalmente projetada para trabalhar no cenário de minereção de preferências *crisp*.

6.1 Método BTF (Xu *et al.*, 2013)

O método de reparação de inconsistência BTF proposto por Xu *et al.* (2013) é um método recursivo, baseado na propriedade de transitividade fraca e tem como principal objetivo, remover todas as inconsistências de uma relação de preferência *fuzzy*.

Rigorosamente falando, o método BTF é dividido em dois algoritmos: o primeiro algoritmo tem a responsabilidade de encontrar todas as inconsistências existentes em uma relação de preferência *fuzzy*. Enquanto o segundo algoritmo é responsável pela reparação dessas inconsistências, tornando a relação de preferência *fuzzy* minimamente consistente com base na transitividade fraca.

A próxima subseção detalha o primeiro algoritmo englobando todos os conceitos necessários para seu entendimento. Seguidamente, a subseção 6.1.2 mostra o funcionamento do segundo algoritmo responsável por reparar as inconsistências.

6.1.1 Encontrando as inconsistências de uma Relação de Preferência *Fuzzy*

Basicamente, encontrar todas as inconsistências de uma relação de preferência *fuzzy*, é equivalente a encontrar todos os ciclos de tamanho 3 no grafo direcionado que representa uma matriz *fuzzy*.

Um ciclo de tamanho 3 significa uma contradição a propriedade de transitividade fraca, pois, cada ciclo encontrado representa uma inconsistência do tipo $m_{ij} \geq 0.5, m_{jk} \geq 0.5 \Rightarrow m_{ik} < 0.5$. Para encontrar os ciclos de tamanho 3, os autores propõem um algoritmo baseado apenas em matrizes e propõem algumas definições como se segue:

Definição 6.1. (Matriz de Adjacência) Seja $M = (m_{ij})_{n \times n}$ uma relação de preferência *fuzzy*, uma matriz de adjacência $E = (e_{ij})_{n \times n}$ sobre M é definida por:

$$e_{ij} = \begin{cases} 1 & \text{se } m_{ij} \geq 0.5, i \neq j; \\ 0 & \text{se } m_{ij} = *; \\ 0 & \text{caso contrario} \end{cases}$$

Onde, "$m_{ij} = *$"representa uma posição vazia da matriz, ou seja, para este par de tuplas o usuário não especificou a preferência.

Exemplo 6.1. Suponha a relação de preferência *fuzzy* M, obtida sobre um conjunto de tuplas $Tup(R) = \{t_1, t_2, t_3, t_4\}$ ilustrada na Figura 6.1(a). A Figura 6.1(b) representa a matriz de adjacência sobre M, obtida pela Definição 6.1. Já a Figura 6.1(c) ilustra o grafo direcionado que representa as preferências da matriz E. A aresta direcionada (t_1,t_4) representa o fato da tupla t_1 ser preferível a tupla t_4, ilustrado na matriz E pelo valor $e_{14} = 1$.

É possível observar no grafo ilustrado na Figura 6.1(c) a existência de dois ciclos de tamanho 3: $c_1 : (t_2 \rightarrow t_1 \rightarrow t_4 \rightarrow t_2)$ e $c_2 = (t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_2)$. Observando matriz M, é notável que cada ciclo representa uma contradição a propriedade de transitividade fraca. Por exemplo, o ciclo c_1 representa as entradas $m_{21} = 0.9$, $m_{14} = 0.7$ e $m_{24} = 0.4$, o que infringe a propriedade de transitividade fraca, pois para que a mesma fosse garantida, $m_{2,4}$ deveria ser maior ou igual a 0.5.

A fim de encontrar todos os ciclos de tamanho 3 a partir da matriz de adjacência E, os autores apresentam duas propostas distintas através do uso de matrizes. Eles defendem

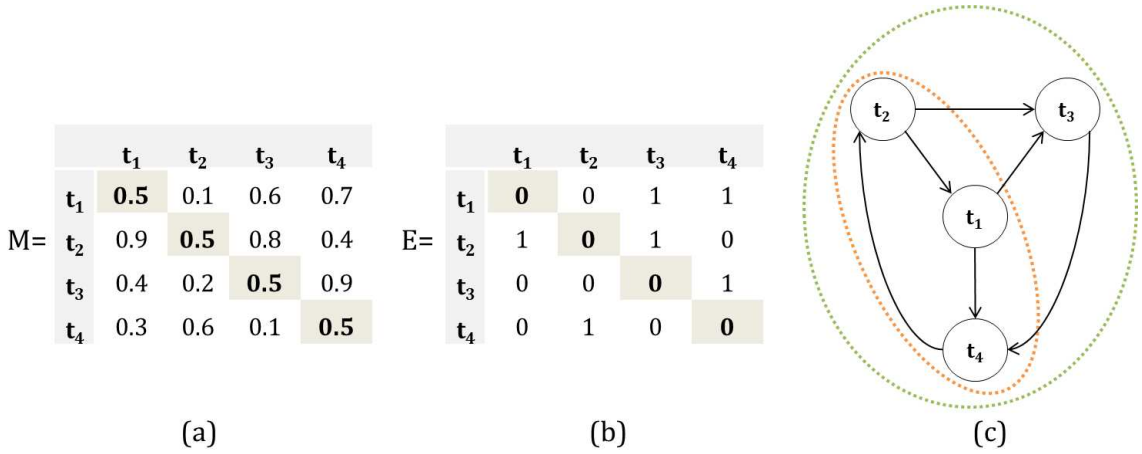


Figura 6.1: (a) Relação de Preferência *Fuzzy* M (b) Matriz de Adjacência E e (c) Grafo Direcionado representando E

que descobrir os ciclos através de matrizes é um processo mais viável quando se tem matrizes de preferência muito grandes.

A primeira proposta para encontrar todos os ciclos de tamanho 3 de uma matriz de preferência *fuzzy* é apenas encontrar a matriz E^3, isto é, a terceira potência da matriz E.

Teorema 6.1. *Seja $M = (m_{ij})_{n\times n}$ uma relação de preferência fuzzy, $E = (e_{ij})_{n\times n}$ a matriz de adjacência de M e E^3 a terceira potência de E. O número de ciclos C de um grafo que representa uma matriz de preferência é dado por:*

$$C = \frac{\sum e_{ij}^3}{3},\ tal\ que,\ i = j \tag{6.1}$$

Ou seja, a soma dos elementos da diagonal principal da matriz E^3 dividido pelo valor três corresponde ao número de ciclos de tamanho 3 de um grafo, que representa a matriz de adjacência de uma relação de preferência *fuzzy*.

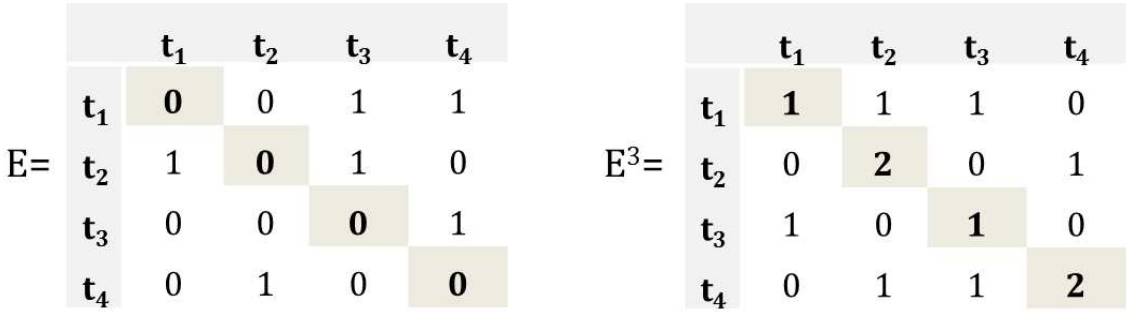


Figura 6.2: Matriz E e Matriz E^3

Continuação do Exemplo 6.1: Considere a matriz E do exemplo anterior e sua terceira potência E^3 ilustrada na Figura 6.2. Calculando o número de ciclos pela Equação 6.1 do Teorema 6.1, é obtido $C = \frac{1+2+1+2}{2} = 2$. Portanto, para a relação de preferência *fuzzy* M ilustrada na Figura 6.1(a) tem-se 2 ciclos de tamanho 3, como ilustrado na Figura 6.1(c).

Outra forma de encontrar o número de ciclos de tamanho 3 de uma matriz de preferência definido em Xu *et al.* (2013), é dada pelo Teorema 6.2.

Teorema 6.2. *Seja $M = (m_{ij})_{n\times n}$ uma relação de preferência fuzzy e $E = (e_{ij})_{n\times n}$ a matriz de adjacência de M. Seja $B = (b_{ij})_{n\times n} = E^2 \circ E^T$ o produto Hadamard de E^2 e E^T, onde E^2 é a segunda potência de E e E^T é a matriz transposta de E.*

Logo, a posição b_{ij} é o número de ciclos de tamanho 3 que incluem o arco direcionado (t_i, t_j). Portanto, $\sum_{j=1}^{n} b_{ij} = e_{ij}^3$ e $\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}$ é três vezes o número de ciclos de tamanho 3 do grafo que representa a matriz E. Ou seja, o número de ciclos C é dado por:

$$C = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}}{3} \tag{6.2}$$

Ao encontrar a matriz B do exemplo anterior, é possível calcular o número de ciclos C. A Figura 6.3 ilustra a obtenção da matriz B. A partir dessa matriz B, é realizado o cálculo do número de ciclos $C = \frac{1+2+1+1+1}{3} = 2$ pela equação 6.2 do Teorema 6.2 . As duas formas de calcular o número de ciclos são eficazes.

E^2=

	t_1	t_2	t_3	t_4
t_1	**0**	1	0	1
t_2	0	**0**	1	2
t_3	0	1	**0**	0
t_4	1	0	1	**0**

E^T=

	t_1	t_2	t_3	t_4
t_1	**0**	1	0	0
t_2	0	**0**	0	1
t_3	1	1	**0**	0
t_4	1	0	1	**0**

B=

	t_1	t_2	t_3	t_4
t_1	**0**	1	0	0
t_2	0	**0**	0	2
t_3	0	1	**0**	0
t_4	1	0	1	**0**

Figura 6.3: Obtenção da matriz B pelo produto *Hadamard* de E^2 e E^T

Após essas definições, o algoritmo para calcular o número de ciclos de tamanho 3 é ilustrado pelo Algoritmo 5. O algoritmo é baseado no Teorema 6.2, pois a matriz B também é utilizada no algoritmo de reparação de inconsistência (Seção 6.1.2).

Algoritmo 5: O Cálculo do Número de Ciclos de Tamanho 3

Entrada: $M = (m_{ij})_n \times n$: uma relação de preferência *fuzzy*.
Saída: O número de ciclos C

1 Encontre a Matriz de Adjacência $E = (e_{ij})_{n\times n}$.
2 Encontre a Matrix $B = (b_{ij})_{n\times n} = E^2 \circ E^T$.
3 Calcule $C = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}}{3}$.
4 **retorne** (C)

Agora, depois de definido o algoritmo que calcula a quantidade de ciclos de tamanho 3, é preciso descrever como funciona o algoritmo de reparação de inconsistência proposto por Xu *et al.* (2013).

6.1.2 Reparando as inconsistências de uma Relação de Preferência *Fuzzy*

Na subseção anterior, foi mostrado que cada valor $b_{ij} \in B$ corresponde ao número de ciclos de tamanho 3 que o arco $(t_i \rightarrow t_j)$ aparece. Um valor $b_{ij} = 4$, significa que o arco $(t_i \rightarrow t_j)$ aparece em quatro ciclos de tamanho 3 diferentes. Logo, são consideradas entradas de inconsistências (e_i), todas as posições da matriz cujo valor $b_{ij} > 0$, pois, é preciso que o arco apareça em pelo menos um ciclo de tamanho 3 para que o mesmo seja considerado uma inconsistência.

Cada arco $(t_i \rightarrow t_j)$ tem seu valor correspondente $m_{ij} \in M$. Esse valor m_{ij} é modificado de acordo com duas regras distintas:

- **Regra 1:** Se um arco inconsistente $(t_i \rightarrow t_j)$ e sua entrada de inconsistência b_{ij} corresponde a entrada de inconsistência que aparece com mais frequência em todos os ciclos, ela deverá ser reparada primeiro.

  Uma entrada de inconsistência com valor muito alto, indica que ela aparece em muitos ciclos, se esta entrada é reparada primeiro, garante-se a eliminação do número máximo de ciclos de uma única vez. Os valores inconsistentes da matriz M corresponde as posições da matriz B cujo valor $b_{ij} > 0$.

- **Regra 2:** Se duas ou mais entradas de inconsistência aparecem com a mesma frequência nos ciclos de tamanho 3, então a entrada m_{ij} com valor mais próximo de 0.5 deverá ser reparada primeiro. Se as distâncias das entradas para o valor 0.5 for a mesma, a escolha de qual entrada será reparada é feita aleatoriamente.

  Uma entrada m_{ij} com valor próximo de 0.5 indica que o usuário teve dificuldade em estabelecer uma preferência entre as duas tuplas e esses casos tendem a inconsistência.

Baseado nas regras 1 e 2 e no Algoritmo 5, o Algoritmo 6 é proposto para reparar a inconsistência de uma relação de preferência *fuzzy*.

Exemplo 6.2. Considere a relação de preferência *fuzzy* M do Exemplo 6.1 ilustrada pela Figura 6.1(a), a matriz de adjacência da Figura 6.1(b) e a matriz B ilustrada na Figura 6.3. Pelo Algoritmo 4 o número de ciclos $C = 2$, logo, é necessário executar o algoritmo de reparação de inconsistência para essa relação de preferência *fuzzy*.

A partir da matriz B todas as entradas de inconsistências são encontradas, isto é, todas as posições (i, j) cujo $b_{ij} > 0$. Neste exemplo, $\mathcal{L}$=$\{b_{12}, b_{2,4}, b_{32}, b_{41}, b_{42}\}$. De $\mathcal{L}$, é preciso encontrar a entrada que aparece com mais frequência, ou seja, $max(\mathcal{L})$=b_{24}, pois $b_{24} = 2$ e as demais entradas possuem valor 1. Neste exemplo, apenas uma entrada de inconsistência aparece com mais frequência, logo, ela deve ser reparada primeiro.

Algoritmo 6: Método BTF

Entrada: $M = (m_{ij})_{n \times n}$: uma relação de preferência *fuzzy*.
Saída: O número de ciclos C e a matriz $M^{(k)}$ minimamente consistente

Faça $k = 0$
Usando o Algoritmo 5 encontre o número de ciclos C.
Faça $M^{(k)} = M$, $E^{(k)} = E$ e $B^{(k)} = B$.
enquanto $C \neq 0$ **faça**
    Encontre o conjunto $\mathcal{L} = max\{b_{ij}\} \in B^{(k)}$
    se $\mathcal{L}$ *possui duas ou mais entradas de inconsistência* **então**
        Escolha a entrada cujo valor m_{ij} correspondente está mais próximo de 0.5
    se *as distâncias entre as entradas e o valor 0.5 forem iguais* **então**
        Escolha entrada de inconsistência aleatoriamente
    Faça $M^{(k+1)} = (m_{ij}^{(k+1)})_{n \times n}$, onde:

$$(m_{ij}^{(k+1)}, m_{ji}^{(k+1)}) = \begin{cases} (1 - m_{ij}^{(k)}, m_{ij}^{(k)}) & \text{se } m_{ij}^{(k)} \text{ é a } e_i \ \mathcal{L} \text{ e } m_{ij}^{(k)} \neq 0.5; \\ (0.4, 0.6) & \text{se } m_{ij}^{(k)} \text{ é a } e_i \ \mathcal{L} \text{ e } m_{ij}^{(k)} = 0.5; \\ (m_{ij}^{(k)}, m_{ji}^{(k)}) & \text{caso contrário} \end{cases}$$

    Faça $k = k + 1$ e usando o Algoritmo 5 encontre o número de ciclos C.
retorne C *e* $M^{(k)}$. $M^{(k)}$ *é relação de preferência fuzzy consistente*

Para efetuar a reparação de inconsistência, é verificado o valor correspondente na matriz M, ou seja, m_{24}. O valor $m_{24} = 0.4$, portanto, a primeira regra para efetuar a reparação é usada, isto é, $(1 - m_{ij}^{(k)}, m_{ij}^{(k)})$ se $m_{ij}^{(k)} \neq 0.5$. Feito a troca, o número de ciclos é calculado novamente. Após essa reparação o número de ciclos $C = 0$, portanto, o algoritmo chega ao fim da execução e a matriz $M^{(k)}$ é retornada.

$$M^{(k)} = \begin{bmatrix} \mathbf{0.5} & 0.1 & \boldsymbol{0.6} & 0.7 \\ 0.9 & \mathbf{0.5} & 0.8 & 0.6 \\ \boldsymbol{0.4} & 0.2 & \mathbf{0.5} & 0.9 \\ 0.3 & 0.4 & 0.1 & \mathbf{0.5} \end{bmatrix}$$

Neste exemplo, nota-se que quando um arco que aparece com mais frequência nos ciclos é escolhido, mais ciclos são eliminados de uma única vez. Nota-se ainda, que o método faz apenas uma troca entre os valores de preferências *fuzzy* com base nas duas regras estabelecidas. Portanto, o método poderá aumentar a qualidade do algoritmo em termos de acurácia, revocação e precisão.

Este método repara todas as inconsistências de uma relações de preferência com base na propriedade de transitividade fraca. O problema encontrado neste método é o tempo de seu processamento, pois trata-se de um algoritmo recursivo que termina sua execução quando o número de ciclos chega ao valor 0. Se existir casos em que, há um número muito grande de ciclos, o método se torna extremamente lento.

6.2 Ranging Voting

Técnicas de votação podem ser utilizadas acopladas à algoritmos de mineração de preferências com o objetivo de melhorar a qualidade do seu modelo. Rigorosamente falando, um algoritmo de mineração de preferências *crisp* pode ter baixa *revocação* quando não consegue comparar muitos pares de tuplas. Para melhorar sua qualidade, a técnica de *Range Voting* (de Amo *et al.*, 2014) pode ser usada para inferir preferências sobre tuplas que o algoritmo não é capaz de comparar.

A técnica de *Range Voting* pode ser aplicada em qualquer algoritmo de mineração de preferências que produza uma função $PREF : Tup \times Tup \rightarrow [0, 1]$, onde Tup é um conjunto de tuplas para os quais se deseja saber a preferência.

Em pesquisas anteriores (de Amo *et al.*, 2012a), (de Amo *et al.*, 2012b), (de Amo *et al.*, 2013), a função $PREF$ associa a um par de tuplas (t, t') o valor 1 (um) caso a tupla t seja preferível a tupla t' e o valor 0 (zero), caso contrário. Neste trabalho, a função $PREF_{rv}$ tem essa mesma definição, no entanto, é acrescentado um módulo para inferir o grau de preferência a partir do resultado dessa função. Por exemplo, para o par (t, t') se a função $PREF$ retorna valor 1, o módulo infere o grau de preferência com valor > 0.5 (indicando que t é preferível a t'). Caso a função $PREF$ retorne valor 0 (zero), o módulo infere o grau de preferência com valor < 0.5 (indica que t' é preferível a t).

A técnica de *Range Voting* funciona baseada na ideia de votação por notas, isto é, dado um conjunto de candidatos, essa técnica verifica qual deles é "eleito"de acordo com as notas dadas a cada um. No contexto desta pesquisa, os candidatos são as tuplas que compõem um banco de dados de preferência *fuzzy* e as notas são os valores retornados pela função $PREF_{rv}$.

O algoritmo *FuzzyPrefMiner* infere o grau de preferência sobre qualquer par de tuplas, uma vez que, quando uma RPB-f não consegue inferir a preferência entre duas tuplas, o componente aleatório se torna responsável por essa tarefa. Portanto, neste algoritmo não existe o problema de pares de tuplas incomparáveis.

O uso da técnica *Range Voting* associada ao algoritmo *FuzzyPrefMiner* visa alcançar dois objetivos principais: (1) diminuir o número de preferências inferidas pelo módulo aleatório e consequentemente(2), inferir relações de preferências *fuzzy* consistentes com base na propriedade de transitividade fraca.

Nas próximas subseções são apresentadas duas versões para essa técnica de votação usando representação *fuzzy*. A primeira versão trata-se de uma versão não incremental, enquanto a segunda versão mostra uma versão incremental da técnica *Range Voting*.

6.2.1 Versão 1 - Range Voting Não-Incremental

A primeira versão da técnica *Range Voting* consiste em verificar entre duas tuplas t e t', qual delas foi mais vezes preferido em relação as outras tuplas do conjunto *Tup* de acordo com a função $PREF_{rv}$ definida pela equação 6.3.

$$PREF_{rv}((t, t', n)) = \frac{\sum_{v} \in \tau(P(t, v) - P(t', v) + 1)}{2 * (n * |\mathcal{T}|)} \tag{6.3}$$

Onde $P(t, v) = \frac{p_1}{p_1+p_2}$ é a razão das probabilidades p_1 e p_2 retornadas pelo FuzzyPrefMiner e $\mathcal{T}$ é o conjunto de tuplas em *Tup*, exceto as tuplas que estão sendo comparados, isto é, t e t'. O Algoritmo 7 ilustra a implementação da função $PREF_{rv}$.

Algoritmo 7: Cálculo da função $PREF_{rv}$

Entrada: *Tup*: um conjunto de tuplas e $((t, t', n))$: uma *tripla.*
Saída: O valor $Pref_{rv}((t, t', n))$

```
1 Faça T = Tup exceto as tuplas t e t'
2 Faça soma = 0
3 para cada t_k ∈ T faça
4     P(t, t_k) = p_1/(p_1 + p_2)
5     P(t', t_k) = p_1/(p_1 + p_2)
6     soma = soma + (P(t, t_k) − P(t', t_k) + 1)
7 Pref_rv(t, t') = soma / (2*(n*|T|))
8 retorne Pref_rv((t, t', n))
```

O resultado retornado pela função $PREF_{rv}$ é um valor dentro do intervalo [0,1] e possui a seguinte interpretação: Se $PREF_{rv} > 0.5$, então a tupla t é preferível a tupla t', se $PREF_{rv} < 0.5$, então a tupla t' é preferível a tupla t e caso $PREF_{rv} = 0.5$, o módulo aleatório é acionado para decidir a preferência. Portanto, o módulo de inferência do grau de preferência irá trabalhar de acordo com o valor retornado pela função $PREF_{rv}$. Essa técnica é acoplada ao algoritmo *FuzzyPrefMiner* e a descrição de sua implementação é descrita no Algoritmo 8.

Após a aplicação da técnica de *Range Voting* Não Incremental em um determinado conjunto de teste, é possível calcular as medidas de avaliação do modelo. A hipótese subjacente, é que com a aplicação desta técnica, o algoritmo consiga comparar mais tuplas sem o módulo aleatório e consequentemente melhorar a sua qualidade. Para compreender como funciona a técnica de *Range Voting* junto com o FuzzyPrefMiner considere um exemplo.

Exemplo 6.3. Suponha um banco de dados de preferência *fuzzy* sobre o conjunto de tuplas $Tup(R) = \{t_1, t_2, t_3, t_4, t_5\}$, ilustrado pela Tabela 6.2 (lado A). Suponha ainda, que na etapa de treinamento do FuzzyPrefMiner o BDP-F foi dividido em 8 faixas de preferência como ilustra a Tabela 6.1.

De acordo com a descrição da técnica de *Range Voting* pelo Algoritmo 8, é necessário computar o conjunto $Tup = Tup_1 \cup Tup_2$ e o produto cartesiano $\mathcal{T}_{\mathrm{X}} = Tup_1 \times Tup_2$. Neste

Algoritmo 8: Range Voting Não Incremental

Entrada: $\mathcal{P}_2$: um banco de dados de preferência *fuzzy* de teste.
Saída: A relação de preferência *fuzzy* predita M_p

```
1  Compute o conjunto Tup_1 de tuplas que aparecem do lado esquerdo de cada tripla em P_2.
2  Compute o conjunto Tup_2 de tuplas que aparecem do lado direito de cada tripla em P_2.
3  Compute o produto cartesiano T_X = Tup_1 × Tup_2.
4  Compute o conjunto Tup = Tup_1 ∪ Tup_2.
5  Aplique o FuzzyPrefMiner sobre T_X.
6  para cada tripla (t, t', n) ∈ T_2 faça
7      Aplique o classificador e descubra a classe λ_k da tripla.
8      Calcule o valor da função PREF_rv((t, t', n)) pelo Algoritmo 7.
9      se PREF_rv((t, t', n)) > 0.5 então
10         gp_predito = λ_k
11     senão se PREF_rv((t, t', n)) < 0.5 então
12         gp_predito = 1 − λ_k
13     senão
14         O preferência é inferida pelo módulo aleatório
15     M_p(t, t') = gp_predito
16 retorne M_p
```

Tabela 6.1: Faixas de Preferências

Faixa	**I=[gp_{min},gp_{max}]**	Classe
c_1	$I_1 = [0.6, 0.65]$	$\lambda_1 = (0.6 + 0.65)/2 = 0.625$
c_2	$I_2 = [0.65, 0.7]$	$\lambda_2 = (0.65 + 0.7)/2 = 0.675$
c_3	$I_3 = [0.7, 0.75]$	$\lambda_3 = (0.7 + 0.75)/2 = 0.725$
c_4	$I_4 = [0.75, 0.8]$	$\lambda_4 = (0.75 + 0.8)/2 = 0.775$
c_5	$I_5 = [0.8, 0.85]$	$\lambda_5 = (0.8 + 0.85)/2 = 0.825$
c_6	$I_6 = [0.85, 0.9]$	$\lambda_6 = (0.85 + 0.9)/2 = 0.875$
c_7	$I_7 = [0.9, 0.95]$	$\lambda_7 = (0.9 + 0.95)/2 = 0.925$
c_8	$I_8 = [0.95, 1.0]$	$\lambda_8 = (0.95 + 1.0)/2 = 0.975$

exemplo, $Tup = Tup(R)$ e $\mathcal{T}_X$ é o próprio BDP-F de teste, pois ele relaciona todas as tuplas.

Os resultados do algoritmo FuzzyPrefMiner são ilustrados pelo lado B da Tabela 6.2. Nesta tabela, é mostrado a classe inferida pelo classificador, o tipo da preferência ("R", se foi inferida por alguma RPB e "A", se foi inferida pelo módulo aleatório), os valores das probabilidades p_1 e p_2 e consequentemente o valor $P(t_i, t_j)$ para facilitar a ilustração do funcionamento da técnica *Range Voting.*

Suponha a *tripla* $1=(t_1, t_2, 0.61)$, cujo FuzzyPrefMiner errou a predição pelo módulo aleatório. Para encontrar o grau de preferência dessa *tripla* pelo *Range Voting* o primeiro passo é encontrar o valor da função $PREF_{rv}(t_1, t_2)$ com relação ao conjunto $\mathcal{T} = \{t_3, t_4, t_5\}$, logo:

$$PREF_{rv}(t, t', n) = \frac{\sum_{v} \in \mathcal{T}^{(P(t,v) - P(t',v) + 1)}}{2 * (n * |\mathcal{T}|)}$$

$$PREF_{rv}(t_1, t_2, 0.61) = P(t_1, t_3) - P(t_2, t_3) + 1 = 0.73 - 0.24 + 1 = 1.49$$

Tabela 6.2: Ilustração de Resultados do FuzzyPrefMiner

A - BPF de Teste				B - Resultados do FuzzyPrefMiner					
Id	t_i	t_j	$\mathbf{gp}_{real}$	**Classe**	**Tipo**	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{gp}_{predito}$	$P(t_i, t_j)$
1	t_1	t_2	0.61	λ_1	A	0.37	0.37	0.375	0.5
2	t_1	t_3	0.73	λ_3	R	0.48	0.18	0.725	0.73
3	t_1	t_4	0.86	λ_6	A	0.25	0.25	0.875	0.5
4	t_1	t_5	0.96	λ_8	R	0.57	0.13	0.975	0.81
5	t_2	t_3	0.64	λ_1	R	0.12	0.37	0.375	0.24
6	t_2	t_4	0.8	λ_3	A	0.17	0.17	0.275	0.5
7	t_2	t_5	0.94	λ_7	R	0.52	0.22	0.925	0.7
8	t_3	t_4	0.69	λ_2	R	0.35	0.15	0.675	0.7
9	t_3	t_5	0.9	λ_7	A	0.28	0.28	0.925	0.5
10	t_4	t_5	0.8	λ_3	R	0.39	0.12	0.725	0.76

$$PREF_{rv}(t_1, t_2, 0.61) = 1.49 + (P(t_1, t_4) - P(t_2, t_4) + 1) = 1.49 + (0.5 - 0.5 + 1) = 2.49$$
$$PREF_{rv}(t_1, t_2, 0.61) = 2.49 + (P(t_1, t_5) - P(t_2, t_5) + 1) = 2.49 + (0.81 - 0.7 + 1) = 3.6$$
$$PREF_{rv}(t_1, t_2, 0.61) = \frac{3.6}{2*(0.61*3)} = \frac{3.6}{3.66} = 0.98$$

Com o valor obtido pela função $PREF_{rv}$, o segundo passo consiste em calcular o grau de preferência inferido. Como $PREF_{rv}(t_1, t_2, 0.61) > 0.5$, é possível inferir que t_1 é preferível a t_2, agora, é necessário calcular o grau de preferência entre as tuplas. A *tripla* foi classificada com classe λ_1, logo, pela linha 10 do Algoritmo 8, o valor $gp_{predito} = \lambda_1 = 0.625$. Esta *tripla* antes comparada pelo FuzzyPrefMiner, indicava que t_2 era preferível a t_1, ou seja, um erro com relação ao BDP-F de teste. Após aplicar a técnica *Range Voting* em todas as tripla do BDP-F, os resultados ilustrados pela Tabela 6.3 são obtidos.

Tabela 6.3: Ilustração dos Resultados da técnica *Range Voting*

Preferências Reais do Usuário				Resultados do *Range Voting*			
Id	t_i	t_j	$\mathbf{gp}_{real}$	**Classe**	**Tipo**	$PREF_{rv}(t_i, t_j)$	$\mathbf{gp}_{predito}$
1	t_1	t_2	0.61	λ_1	R	0.98	0.625
2	t_1	t_3	0.73	λ_3	R	0.65	0.725
3	t_1	t_4	0.86	λ_6	R	0.67	0.875
4	t_1	t_5	0.96	λ_8	R	0.64	0.975
5	t_2	t_3	0.64	λ_1	R	0.84	0.625
6	t_2	t_4	0.8	λ_3	R	0.6	0.725
7	t_2	t_5	0.94	λ_7	R	0.59	0.925
8	t_3	t_4	0.69	λ_2	R	0.67	0.675
9	t_3	t_5	0.9	λ_7	R	0.74	0.925
10	t_4	t_5	0.8	λ_3	R	0.69	0.725

Neste exemplo, é notável que com a aplicação do *Range Voting* o porcentagem de acertos dos graus de preferências aumenta em relação aos resultados obtidos somente pelo FuzzyPrefMiner. A porcentagem de acertos do FuzzyPrefMiner é de 60% (incluindo inferências pela RPB e pelo módulo aleatório), enquanto a porcentagem de acertos do *Range Voting* é de 90% (diminuindo o número de preferências inferidas pelo módulo

aleatório). Outra evidência que se pode notar neste exemplo, é que o número de ciclos de tamanho 3 (representa inconsistência) da matriz resultante pelo FuzzyPrefMiner também diminui de 2 (dois ciclos) para 0 (zero) após a aplicação do *Range Voting*.

Pode-se notar, que a técnica *Range Voting* Não Incremental utiliza as inferências feitas pelo algoritmo de mineração de preferências para reforçar as preferências sobre os pares de tuplas. Logo, é possível inferir preferências sobre pares de tuplas que o algoritmo descobriu as preferências pelo módulo aleatório, além de corrigir inferências feitas de forma incorreta pelo algoritmo. Na próxima subseção, uma versão incremental da técnica *Range Voting* é apresenta. Esta nova versão tende a ser mais promissora que esta versão Não Incremental.

6.2.2 Versão 2 - Ranging Voting Incremental

A ideia subjacente à técnica *Range Voting* Incremental é semelhante a versão Não-Incremental apresentada na subseção anterior. A principal diferença desta segunda versão, consiste em atualizar o valor $P(t, t')$ pelo valor retornado pela função $PREF_{rv}(t, t', n)$. Ou seja, a medida que é calculado o valor da função $PREF_{rv}$ para uma determinada *tripla* (t, t', n), o valor $P(t, t')$ retornado inicialmente pela razão das probabilidades p_1 e p_2 é atualizado pelo valor $PREF_{rv}$.

Rigorosamente falando, o objetivo desta segunda versão assim como a versão anterior é diminuir o número de inferências pelo módulo aleatório do FuzzyPrefMiner, aumentar o número de graus de preferências inferidos corretamente e consequentemente diminuir o número de ciclos da matriz de preferência predita.

A hipótese é que esta segunda versão tenha melhor desempenho que a versão não incremental, principalmente quando aplicado em grandes quantidades de dados. A descrição da implementação do *Range Voting* Incremental pode ser vista com mais detalhes no Algoritmo 9.

Nesta versão incremental da técnica *Range Voting* é considerado o conjunto de *triplas* cujo $P(t, t') \geq 0.5$, ou seja, são consideradas aquelas *triplas* que tem a preferência mais certa possível. Além disso, na linha 7 do algoritmo 9 nota-se uma ordenação do conjunto $\mathcal{T}_{+0.5}$, isto é, as *triplas* mais incertas são consideradas antes que as mais certas.

Na linha 11, o valor $P(t, t')$ é incrementado pelo valor retornado pela função $PREF_{rv}$. Assim, quando a técnica é aplicada para a segunda *tripla* do conjunto $\mathcal{T}_{+0.5}$, já não é considerado valor inicial $P(t, t')$ retornado pelo FuzzyPrefMiner, mas o valor $PREF_{rv}$ retornado pelo *Range Voting*.

Por exemplo, considere a *tripla* $2 = (t_1, t_3, 0.73)$ da Tabela 6.2 ilustrada no Exemplo 6.3. Para calcular a função $PREF_{rv}$ pelo *Range Voting* Incremental desta *tripla* não é levado em consideração o valor $P(t_1, t_2) = 0.5$, mas sim o valor $P(t_1, t_2) = 0.98$, pois, $P(t_1, t_2)$ é atualizado pelo valor retornado pela função $PREF_{rv}(t_1, t_2, 0.61)$.

Algoritmo 9: Range Voting Incremental

Entrada: $\mathcal{P}_2$: um banco de dados de preferência *fuzzy* de teste.
Saída: A relação de preferência *fuzzy* predita M_p

```
1  Compute o conjunto Tup_1 das tuplas que aparecem do lado esquerdo de cada tripla em P_2.
2  Compute o conjunto Tup_2 das tuplas que aparecem do lado direito de cada tripla em P_2.
3  Compute o produto cartesiano T_x = Tup_1 × Tup_2.
4  Compute o conjunto Tup = Tup_1 ∪ Tup_2.
5  Aplique o FuzzyPrefMiner sobre T_x.
6  Compute T_{+0.5} = todas as triplas em T_x cujo P(t,t') ≥ 0.5.
7  Ordene T_{+0.5} por ordem crescente de P(t,t')
8  para cada tripla (t,t',n) ∈ T_{+0.5} faça
9      Aplique o classificador e descubra a classe λ_k da tripla.
10     Calcule o valor da função PREF_rv(t,t',n) pelo Algoritmo 7.
11     Atualize o valor P(t,t') = PREF_rv(t,t',n)
12     se (t,t',n) ∈ P_2 então
13         se PREF_rv(t,t',n) > 0.5 então
14             gp_predito = λ_k
15         senão se PREF_rv(t,t',n) < 0.5 então
16             gp_predito = 1 − λ_k
17         senão
18             A preferência é inferida aleatoriamente
19         M_p(t,t') = gp_predito
20 retorne M_p
```

Portanto, o cálculo desta função para a *tripla* $(t_1, t_3, 0.73)$ com relação ao conjunto com relação ao conjunto $\mathcal{T} = \{t_2, t_4, t_5\}$ é dado por:

$$PREF_{rv}(t, t', n) = \frac{\sum_{v} \in \mathcal{T}(P(t,v) - P(t',v) + 1)}{2 * (n * |\mathcal{T}|)}$$

$$PREF_{rv}(t_1, t_3, 0.73) = P(t_1, t_2) - P(t_3, t_2) + 1 = 0.98 - 0.76 + 1 = 1.22$$

$$PREF_{rv}(t_1, t_3, 0.73) = 1.22 + (P(t_1, t_4) - P(t_3, t_4) + 1) = 1.22 + (0.5 - 0.7 + 1) = 2.02$$

$$PREF_{rv}(t_1, t_3, 0.73) = 2.02 + (P(t_1, t_5) - P(t_3, t_5) + 1) = 2.02 + (0.81 - 0.5 + 1) = 3.33$$

$$PREF_{rv}(t_1, t_3, 0.73) = \frac{3.33}{2 * (0.73 * 3)} = \frac{3.6}{4.38} = 0.76$$

Na versão Não Incremental, o valor retornado pela função $PREF_{rv}(t_1, t_3, 0.73) = 0.65$ (veja Tabela 6.3 lado B), enquanto a versão incremental retorna $PREF_{rv}(t_1, t_3, 0.73) = 0.76$. Para este exemplo, o grau de preferência predito não mudaria, mas, suponha um caso em que o *Range Voting* Não Incremental retorne um valor $PREF_{rv} < 0.5$, inferindo um valor de grau de preferência de forma incorreta ou pelo módulo aleatório (caso $PREF_{rv} = 0.5$), nestes casos, o *Range Voting* Incremental é uma opção interessante para obter melhores resultados.

De um modo geral, essa segunda versão da técnica *Range Voting* é um refinamento da primeira versão apresentada na Subseção 6.2.1. Este refinamento tem como principal objetivo, inferir preferências sobre os pares de tuplas que a primeira versão não consegue comparar ou compara de forma incorreta.

6.3 Considerações do Capítulo

Neste capítulo foi apresentado o método proposto por Xu *et al.* (2013) para reparação de inconsistência para relações de preferência *fuzzy*. Este métoto tem como objetivo é reparar toda a inconsistência de uma relação de preferência *fuzzy*. A apresentação em detalhes deste método se fez necessária por duas razões: (1) O algoritmo que avalia a inconsistência das relações de preferências *fuzzy* é utilizado nesta dissertação e (2) este trabalho (Xu *et al.*, 2013) é o *baseline* de comparação escolhido para validar os métodos de reparação de inconsistência propostos nesta dissertação.

A última seção deste capítulo apresentou a segunda contribuição deste trabalho, isto é, destacou os dois novos métodos baseados na técnica *Range Voting* (de Amo *et al.*, 2014) para trabalhar juntamente com o algoritmo FuzzyPrefMiner, a fim de produzir uma relação de preferência *fuzzy* consistente. No cenário *criso*, essa técnica melhora a precisão do algoritmo de mineração de preferências e diminui a quantidade de pares de tuplas que o algoritmo não consegue comparar aumentando a revocação do mesmo. O objetivo do uso desta técnica no cenário de mineração de preferências *fuzzy* é melhorar a qualidade do algoritmo FuzzyPrefMiner e predizer relações de preferências *fuzzy* consistentes.

Capítulo 7

Resultados Experimentais

Dividido em três seções, este capítulo tem como objetivos: (1) Apresentar os dados reais utilizados nos experimentos, (2) apresentar os resultados obtidos pelo algoritmo de mineração de preferências proposto nesta dissertação e finalmente, (3) mostrar os resultados obtidos pelos métodos de reparação de inconsistência em relações de preferências *fuzzy*.

Este capítulo visa validar a principal hipótese que guiou este trabalho de dissertação: *O grau de preferência é um fator importante para a qualidade do modelo de preferências minerado.*

Dessa forma, a Seção 7.1 apresenta os dados reais utilizados em testes e detalha a técnica de amostragem utilizada no pré-processamento dos dados. A Seção 7.2 apresenta em detalhes os resultados obtidos pelo Algoritmo *FuzzyPrefMiner*, bem como, a comparação com o algoritmo *CPrefMiner* (de Amo *et al.*, 2012a). A Seção 7.3 mostra os resultados obtidos pela técnica *Range Voting* e o Método BTF (Xu *et al.*, 2013), utilizados para reparar relações de preferências *fuzzy* inconsistentes inferidas pelo FuzzyPrefMiner.

7.1 Dados de Teste e Técnica de Amostragem

Com o objetivo de avaliar a qualidade do Algoritmo *FuzzyPrefMiner* proposto nesta dissertação, seis bancos de dados reais contendo preferências sobre filmes são considerados. Estes dados disponibilizados pelo projeto *GroupLens* [1]contém avaliações sobre filmes.

As avaliações dos filmes têm disponível a identificação do usuário (ID-Usuário), a identificação do filme (ID-Filme) e a nota (N) que o usuário deu para o filme. Para coletar

[1]www.grouplens.org

as demais características do filme, tais como: Gênero, Ator, Idioma, Ano e Diretor foi criado um *crawler* que extrai essas características do site IMDb [2].

Os dados escolhidos para realizar todos os experimentos possuem as características descritas na Tabela 7.1. Estes seis banco dados foram considerados e cada banco corresponde a um usuário que avaliou uma certa quantidade de filmes. Os usuários foram escolhidos de forma que, o segundo tenha avaliado aproximadamente 3× mais filmes que o primeiro usuário e assim sucessivamente.

Tabela 7.1: Bancos de Dados Escolhidos para Experimentos

BDP-F	**# Filmes Avaliados**	**# *Triplas***
D_1	300	34.526
D_2	610	146.804
D_3	996	336.399
D_4	1214	578.426
D_5	1454	836.949
D_6	1688	1.106.029

Uma vez escolhido os dados para realização dos experimentos, se faz necessária a escolha de uma técnica de amostragem de dados para validação do Modelo de Preferência extraído pelo *FuzzyPrefMiner*.

A técnica de amostragem utilizada para avaliar a qualidade do modelo de preferência é a *k-fold-cross-validation* ou *validação cruzada* com valor $k = 30$. Nesta técnica de amostragem, um conjunto de dados D_i é dividido em k subconjuntos $\{D_{i_1}, D_{i_2} \dots D_{i_k}\}$ de aproximadamente mesmo tamanho.

O modelo portanto é treinado e testado k vezes. Em cada tempo $t \in \{1, \dots, k\}$, $k-1$ subconjuntos são usados para treinamento do modelo e 1 subconjunto é usado para teste. A qualidade do modelo é avaliada a cada tempo t e ao final da execução dos k tempos, o valor médio da medida de qualidade é retornado.

Com o objetivo de dividir o conjunto de dados em subconjuntos homogêneos foi utilizado um protocolo de estratificação dos dados. Este protocolo denominado *Weak Protocol* foi introduzido em de Amo *et al.* (2013) e a principal ideia subjacente a este protocolo, é que a porcentagem de *triplas* com determinado grau de preferência em cada partição seja a mesma do conjunto global de dados.

7.2 Validação da hipótese da relevância do grau de preferência

Nesta seção são apresentados os resultados obtidos pelo algoritmo *FuzzyPrefMiner* quando aplicado tanto no cenário *crisp*, quanto no cenário *fuzzy*. O FuzzyPrefMiner

[2] imdb.com

foi implementado na linguagem de programação Java e todos os experimentos foram realizados em uma máquina com processador *core* i7 3.20GHZ com 32GB de memória RAM rodando sob o sistema operacional Linux Ubuntu 12.10.

Os seguintes valores de parâmetros foram considerados na execução de todos os testes: (1) Procedimento "*Extração*"representado pelo Algoritmo 3 : $\beta = 50$, $\vartheta = 100$ e $\gamma = 20$. No Algoritmo 4 (cálculo do grau de dependência entre os atributos): $\alpha_1 = 0.2$ e $\alpha_2 = 0.2$.

7.2.1 Grupo de Teste 1: Escolha do Classificador

Teste 1.1: Para escolher o classificador que melhor se adequa aos dados de filmes foi realizado um pequeno teste. Neste teste, primeiro foi criado o BDP-F com 31713 triplas. A partir desse banco de preferência *fuzzy* foi criado um arquivo *Weka* (.arff), tal que, as *triplas* desse arquivo são classificadas em 4 classes de acordo com seu grau de preferência: (1) Classe λ_1 com intervalo $I_1 = (0.6, 0.7]$; (2) Classe λ_2 com intervalo $I_2 = (0.7, 0.8]$; (3) Classe λ_3 com intervalo $I_2 = (0.8, 0.9]$ e (4) Classe λ_4 com intervalo $I_4 = (0.9, 1.0]$.

Como descrito no Capítulo 5, o classificador é utilizado para predizer a "intensidade de preferência"entre duas tuplas distintas. Esta intensidade pode ser tanto positiva (>0.5), quanto negativa (<0.5).

Vários classificadores foram aplicados com a ferramenta *Weka* com a técnica de amostragem padrão *10-cross-validation* e os resultados mostrados na Tabela 7.2 foram obtidos.

Tabela 7.2: Acurácia dos Classificador com Ferramenta *Weka*

Classificador	**Acurácia**
AdaBoostM1	25.37%
OneR	55.05%
AttributeSelectClassifier	56.48%
REPTree	57.49%
Ibk	59.41%
J48	60%
DeciosionTable	61.3%
Kstar	64.93%
PART	74.68%
NaiveBayes	79.97%
BayesNet	**80.25%**

Os classificadores *bayesianos* (*NaiveBayes* e *BayesNet*) se mostraram mais performáticos que os demais classificadores utilizados neste teste. Entretanto, o protocolo de estratificação utilizado pela ferramenta *Weka* não é o mesmo utilizado pelo algoritmo *FuzzyPrefMiner*. Portanto, um segundo teste foi executado para auxiliar na escolha do classificador.

Teste 1.2: Com este segundo teste envolvendo diferentes classificadores, o objetivo é analisar a influência de diferentes classificadores quando aplicados juntamente com o algoritmo *FuzzyPrefMiner* no cenário *Crisp*.

Foram escolhidos três classificadores distintos com base nos resultados obtidos no Teste 1.1, são eles: (1) o *AdaBoosM1* (que retornou o pior resultado da Tabela 7.2); (2) o *DecisionTable* (que retornou um resultado mediano) e (3) o classificador *BayesNet* (retornou o melhor resultado).

Tabela 7.3: Influência dos Classificadores: *acc*, *revoc* e *prec*

BDP-F	***AdaBoostM1***			***DecisionTable***			***BayesNet***		
	acc	*revoc*	*prec*	*acc*	*revoc*	*prec*	*acc*	*revoc*	*prec*
D_1	75.84%	74.06%	76.73%	77.29%	75.40%	78.39%	**82.48%**	**80.76%**	**83.67%**
D_2	73.49%	71.86%	74.26%	77.53%	75.39%	78.73%	**83.29%**	**81.99%**	**84.20%**
D_3	78.48%	73.48%	81.65%	78.25%	76.65%	79.14%	**87.49%**	**86.39%**	**88.33%**
D_4	72.82%	72.14%	73.14%	76.47%	77.07%	77.23%	**86.39%**	**85.71%**	**86.91%**
D_5	79.09%	75.81%	81.15%	75.29%	73.02%	76.50%	**86.10%**	**85.34%**	**86.66%**
D_6	73.77%	72.95%	74.17%	79.36%	78.41%	79.93%	**83.80%**	**82.98%**	**84.35%**

O algoritmo *FuzzyPrefMiner* foi aplicado no cenário *Crisp*, variando esses três classificadores com quatro faixas de preferências como descrito no Teste 1.1 e os resultados mostrados na Tabela 7.3 foram obtidos.

Mesmo com uma estratificação diferente da utilizada pela ferramenta *Weka*, o classificador *BayesNet* obteve melhores resultados de acurácia, revocação e precisão em todos os conjuntos de dados.

Tabela 7.4: Influência dos Classificadores: *tc* e *ta*

BDP-F	***AdaBoostM1***		***DecisionTable***		***BayesNet***	
	tc	*ta*	*tc*	*ta*	*tc*	*ta*
D_1	95.52%	1.78%	96.18%	1.90%	**96.53%**	**1.72%**
D_2	96.89%	1.53%	95.76%	2.13%	**97.38%**	**1.30%**
D_3	89.99%	5.01%	**98.85%**	1.60%	97.80%	**1.09%**
D_4	**98.63%**	0.69%	97.21%	1.40%	98.62%	**0.68%**
D_5	93.43%	3.28%	95.45%	2.26%	**98.48%**	**0.76%**
D_6	98.35%	0.83%	98.10%	0.95%	**98.38%**	**0.81%**

Quando se trata das medidas de *Taxa de Comparabilidade* e *Taxa de Aleatoriedade* os resultados (veja Tabela 7.4) se mostram bastantes semelhantes na variação dos classificadores. Portanto, para os demais experimentos descritos nesta seção o classificador *BayesNet* é considerado.

7.2.2 Grupo de Teste 2: Influência das Faixas de Preferências

O segundo grupo de teste tem como objetivo, avaliar o desempenho do algoritmo *FuzzyPrefMiner* no cenário *Crisp*, variando o número de faixas de preferências para o particionamento de um BDP-F.

Para determinar as faixas de preferências utilizadas neste grupo de teste, o comportamento da função de transformação foi observado (Equação 2.2, Capítulo 2) quando aplicada sobre as notas dos filmes avaliados pelo usuário. Ao aplicar a função, foi observado que o menor grau de preferência $gp_{min} = 0.6097$ (obtido pela função de transformação sobre as notas 5 e 4) e o maior grau de preferência é $gp_{max} = 0.9615$ (obtido pela função de transformação sobre as notas 5 e 1), logo, o intervalo $I = (0.6, 1.0)$ foi considerado para realizar a divisão de faixas de preferências. Dessa forma, os graus de preferências são distribuídos de acordo com as faixas de preferências ilustradas na Tabela 7.5.

Tabela 7.5: Faixas de Preferências

k=2	**k=4**	**k=8**
		$I_1 = (0.6,0.65]$
		$I_2 = (0.65,0.7]$
	$I_1 = (0.6,0.7]$	$I_3 = (0.7,0.75]$
$I_1 = (0.6,0.8]$	$I_2 = (0.7,0.8]$	$I_4 = (0.75,0.8]$
$I_2 = (0.8,1.0]$	$I_3 = (0.8,0.9]$	$I_5 = (0.8,0.85]$
	$I_4 = (0.9,1.0]$	$I_6 = (0.85,0.9]$
		$I_7 = (0.9,0.95]$
		$I_8 = (0.95,1.0]$

Com a divisão do BDP-F em faixas de preferências, é possível analisar a influência dessas faixas na criação do modelo de preferências. Os resultados de acurácia, revocação e precisão são mostrados na Tabela 7.6. Como esperado, quanto mais refinada as faixas de preferências ($k = 8$), melhores são os resultados obtidos.

Tabela 7.6: Influência das Faixas de Preferências (*acc*, *revoc* e *prec*)

BDP-F	k=2			k=4			k=8		
	acc	*revoc*	*prec*	*acc*	*revoc*	*prec*	*acc*	*revoc*	*prec*
D_1	75.59%	73.23%	76.72%	82.48%	80.76%	83.67%	**87.41%**	**85.64%**	**88.80%**
D_2	74.96%	73.28%	75.84%	83.29%	81.98%	84.20%	**88.42%**	**87.17%**	**89.45%**
D_3	72.27%	70.71%	72.97%	87.49%	86.39%	88.33%	**90.09%**	**89.21%**	**90.83%**
D_4	72.84%	72.14%	73.16%	86.39%	85.71%	86.91%	**89.96%**	**89.23%**	**90.56%**
D_5	73.98%	73.10%	74.40%	86.11%	85.34%	86.66%	**89.25%**	**88.50%**	**89.85%**
D_6	73.04%	72.31%	73.37%	83.80%	82.98%	84.35%	**89.48%**	**88.70%**	**90.07%**

Tabela 7.7: Influência das Faixas de Preferências (*tc* e *ta*)

BDP-F	k=2		k=4		k=8	
	tc	*ta*	*tc*	*ta*	*tc*	*ta*
D_1	95.46%	2.35%	**96.53%**	**1.72%**	96.45%	1.76%
D_2	96.62%	1.68%	97.38%	1.30%	**97.46%**	**1.25%**
D_3	96.91%	1.56%	97.80%	1.09%	**98.21%**	**0.89%**
D_4	98.61%	0.69%	**98.62%**	**0.68%**	98.53%	0.73%
D_5	98.26%	0.87%	98.48%	0.76%	**98.49%**	**0.75%**
D_6	**98.56%**	**0.73%**	98.37%	0.81%	98.48%	0.77%

Quando se trata das Taxas de Comparabilidade e Aleatoriedade do algoritmo, nota-se, que os resultados não tem diferenças muito significantes com respeito a variação do número de faixas de preferências como ilustra a Tabela 7.7. Os resultados de todos os valores k são satisfatórios, pois, tem-se alta taxa de comparabilidade (em torno de 98%) e baixos valores de taxa de aleatoriedade (abaixo de 2.5%). Portanto, o método *FuzzyPrefMiner* tem boa predição, pois a maioria dos acertos são obtidos pela ordem de preferência da RPB-f e não pelo módulo aleatório.

7.2.3 Grupo de Teste 3: Comparação do FuzzyPrefMiner no cenário *Crisp* e cenário *Fuzzy*

O terceiro grupo de teste objetiva analisar o desempenho do algoritmo FuzzyPrefMiner quando aplicado nos cenários *crisp* e *fuzzy*. A aplicação do FuzzyPrefMiner no cenário *crisp*, consiste em predizer apenas qual a tupla preferida entre duas tuplas distintas. Enquanto sua aplicação no cenário *fuzzy*, visa predizer o grau de preferência entre duas tuplas. Para o algoritmo FuzzyPrefMiner aplicado no cenário *fuzzy*, foi considerado o parâmetro $\kappa = 0.05$ para analisar os acertos da predição.

Para comparar o algoritmo aplicado nos dois cenários, foi usado o teste de significância estatística ***unicaudal*** descrito em Urdan (2010) (usado para comparar o desempenho de classificadores) adaptado ao cenário de mineração de preferências.

Para cada uma das medidas de qualidade d', tal que, $d' \in \{acc, revoc, prec, tc, ta\}$, seja $\bar{d'}$ a diferença média entre a medida d' do algoritmo FuzzyPrefMiner aplicado no cenário *crisp* e a respectiva medida d' do algoritmo aplicado no cenário *fuzzy*. O objetivo deste teste de significância estatística, é calcular o intervalo de confiança γ para a medida d' e testar se a Hipótese Nula é rejeitada. Este intervalo de confiança é calculado pela fórmula: $\gamma = t_{(1-\alpha,k-1)} \times \sigma_{d'}$, onde $t_{(1-\alpha,k-1)}$ é um coeficiente obtido pela tabela de distribuição t, tal que, $(1-\alpha)$ é o nível de confiança e $(k-1)$ o grau de liberdade. Nestes experimentos, foram considerados $(1-\alpha) = 95$ e $k = 30$ e a significância estatística foi calculada para todas as medidas de avaliação, uma vez que, todas foram avaliadas em amostras de mesmo tamanho.

A pergunta de interesse neste experimento é: *Com 95% de certeza, é possível concluir que o FuzzyPrefMiner no cenário Crisp supera os resultados obtidos no cenário Fuzzy*? Portanto, para este experimento dois casos devem ser analisados para as diferentes medidas de avaliação.

Caso 1: As hipóteses nula e alternativa para as medidas *acc*, *revoc*, *prec* e *tc* são:

- H0: FuzzyPrefMiner – *Crisp* $\leq$ FuzzyPrefMiner – *Fuzzy*
- HA: FuzzyPrefMiner – *Crisp* $>$ FuzzyPrefMiner – *Fuzzy*

Caso 2: As hipóteses nula e alternativa para a medida de *ta* são:

- H0: FuzzyPrefMiner – *Crisp* $\geq$ FuzzyPrefMiner – *Fuzzy*
- HA: FuzzyPrefMiner – *Crisp* $<$ FuzzyPrefMiner – *Fuzzy*

Rigorosamente falando, para que o FuzzyPrefMiner no cenário *Crisp* seja superior ao cenário *fuzzy*, é preciso que os resultados obtidos no cenário *crisp* sejam superiores nas medidas de *acc*, *revoc*, *prec* e *tc*, e para medida de avaliação *ta*, é necessário que o resultado obtido pelo FuzzyPrefMiner – *Crisp* seja inferior. Portanto, é preciso que a diferença entre essas medidas seja estatisticamente significante e que a Hipótese Nula seja rejeitada.

No Caso 1, para que a hipótese nula seja rejeitada, os valores do intervalo de confiança γ da medida $d' \in \{acc, revoc, prec, tc\}$ devem ser positivos e para que a diferença seja considerada estatisticamente significante, o valor 0 não deve ser abrangido por esse intervalo. Os resultados ilustrados pela Tabela 7.8 mostram que a diferença observada para as medidas de *acc*, *revoc* e *prec* é estatisticamente significante, pois todos os valores do intervalo de confiança dessas medidas são positivos e não abrangem o valor 0, portanto, para essas três medidas, a hipótese nula é rejeitada e a hipótese alternativa é substanciada.

Tabela 7.8: FuzzyPrefMiner: *Crisp* $\times$ *Fuzzy* (*acc*, *revoc* e *prec*)

BDP-F	FuzzyPrefMiner: *Crisp*			FuzzyPrefMiner: *Fuzzy*					
	acc	*revoc*	*prec*	*acc*	γ	*revoc*	γ	*prec*	γ
D_1	**87.41%**	**85.64%**	**88.80%**	81.60%	0.0004	79.93%	0.0004	82.77%	0.0004
D_2	**88.42%**	**87.17%**	**89.45%**	85.90%	0.0002	84.73%	0.0002	86.96%	0.0002
D_3	**90.09%**	**89.21%**	**90.83%**	88.33%	0.0001	87.41%	0.00009	89.13%	0.00008
D_4	**89.96%**	**89.23%**	**90.56%**	88.16%	0.0001	87.44%	0.00009	88.76%	0.0001
D_5	**89.25%**	**88.50%**	**89.86%**	86.97%	0.00009	86.18%	0.00008	87.65%	0.00008
D_6	**89.48%**	**88.70%**	**90.07%**	86.96%	0.0001	86.18%	0.0001	87.58%	0.0001

Quando se trata da medida de avaliação *tc* (veja Tabela 7.9), apenas no primeiro usuário têm-se valores negativos no intervalo de confiança, logo, para o BDP-F D_1 a hipótese nula é mantida e a hipótese alternativa é rejeitada. Para os demais BDP-Fs a hipótese nula é rejeitada e a alternativa é substanciada. E como, o valor 0 não é abrangido pelos intervalos, a diferença observada entre o FuzzyPrefMiner no cenário *crisp* e cenário *fuzzy* é considerada estatisticamente significante.

Tabela 7.9: FuzzyPrefMiner: *Crisp* $\times$ *Fuzzy* (*tc* e *ta*)

BDP-F	FuzzyPrefMiner: *Crisp*		FuzzyPrefMiner: *Fuzzy*			
	tc	*ta*	*tc*	γ	*ta*	γ
D_1	96.45%	1.76%	**96.57%**	0,0002	**1.67%**	0,0003
D_2	**97.46%**	1.25%	97.43%	0,0001	**1.17%**	0,0001
D_3	**98.21%**	**0.89%**	98.07%	0,00003	0.92%	0,00011
D_4	**98.53%**	0.73%	98.51%	0,00003	**0.72%**	0,00004
D_5	**98.49%**	**0.75%**	98.32%	0,00002	0.79%	0,00013
D_6	**98.48%**	**0.77%**	98.40%	0,00003	0.78%	0,00005

No Caso 2, para que a hipótese nula seja rejeitada é necessário que os valores do intervalo de confiança da medida *ta* não tenha valores positivos. Na Tabela 7.9, apenas

nos BDP-Fs D_3 e D_5 a hipótese nula é rejeitada, nos demais banco de dados a hipótese nula é mantida.

Com estes testes, é possível concluir que o algoritmo FuzzyPrefMiner quando aplicado no cenário *crisp*, consegue melhores resultados quando sua aplicação é executada no cenário *fuzzy*. Este resultado, de certa forma era o esperado, pois o algoritmo quando aplicado no cenário *fuzzy* envolve predição de resultados mais refinados, como o grau de preferência.

7.2.4 Grupo de Teste 4: Comparação com o algoritmo CPrefMiner

O quarto grupo de teste tem como objetivo, avaliar a influência do grau de preferência no desempenho do algoritmo FuzzyPrefMiner. Este grupo de experimentos visa validar a hipótese original deste trabalho de mestrado: "*quando se tem mais informações na entrada dos dados, melhores são os resultados de desempenho do algoritmo de mineração de preferências*".

Com o objetivo de validar essa hipótese, o algoritmo FuzzyPrefMiner (tanto no cenário *crisp*, quanto no cenário *fuzzy*) é comparado com o algoritmo CprefMiner (de Amo *et al.*, 2012a), descrito no Capítulo 4 e desenvolvido para solucionar apenas o problema de mineração no cenário *crisp*.

Foi aplicado o mesmo teste de significância estatística descrito no Grupo de Teste 3 da seção anterior. Portanto, a pergunta de interesse deste grupo de experimento é: *Com 95% de certeza, é possível concluir que o FuzzyPrefMiner supera os resultados obtidos pelo CPrefMiner?* Logo, para as medidas avaliadas neste grupo de experimento também têm-se dois casos:

Caso 1: As hipóteses nula e alternativa para as medidas *acc*, *revoc*, *prec* e *tc* são:

- H0: FuzzyPrefMiner – *Crisp* $\leq$ FuzzyPrefMiner – *Fuzzy*
- HA: FuzzyPrefMiner – *Crisp* $>$ FuzzyPrefMiner – *Fuzzy*

Caso 2: As hipóteses nula e alternativa para a medida de *ta* são:

- H0: FuzzyPrefMiner – *Crisp* $\geq$ FuzzyPrefMiner – *Fuzzy*
- HA: FuzzyPrefMiner – *Crisp* $<$ FuzzyPrefMiner – *Fuzzy*

A Tabela 7.10 mostra os resultados obtidos pelo FuzzyPrefMiner no cenário *crisp* em comparação com o *baseline* CPrefMiner. Pode-se notar, que para as medidas de avaliação *acc*, *revoc* e *prec* a hipótese nula é rejeitada (Caso 1) e a hipótese alternativa (Caso 1) é substanciada. Os intervalos de confiança mostram que as diferenças observadas nestas três medidas são estatisticamente significantes.

Tabela 7.10: FuzzyPrefMiner -*Crisp* × CPrefMiner (*acc, revoc* e *prec*)

BDP-F	FuzzyPrefMiner - *Crisp*			CPrefMiner					
	acc	*revoc*	*prec*	*acc*	γ	*revoc*	γ	*prec*	γ
D_1	**87.41%**	**85.64%**	**88.80%**	81.24%	0,001	79.46%	0,0012	82.34%	0,001
D_2	**88.42%**	**87.17%**	**89.45%**	80.48%	0,0012	79.46%	0,001	81.18%	0,0014
D_3	**90.09%**	**89.21%**	**90.83%**	83.86%	0,0018	83.23%	0,0016	84.33%	0,0019
D_4	**89.96%**	**89.23%**	**90.56%**	81.80%	0,0023	80.45%	0,0028	82.72%	0,0023
D_5	**89.25%**	**88.50%**	**89.86%**	81.37%	0,0027	80.01%	0,0031	82.25%	0,0028
D_6	**89.48%**	**88.70%**	**90.07%**	81.30%	0,0014	80.86%	0,0013	81.57%	0,0014

Tabela 7.11: FuzzyPrefMiner: *Crisp* × CPrefMiner (*tc* e *ta*)

BDP-F	FuzzyPrefMiner - *Crisp*		CPrefMiner			
	tc	*ta*	*tc*	γ	*ta*	γ
D_1	96.45%	**1.76%**	**96.51%**	0,0012	1.78%	0,0007
D_2	97.46%	1.25%	**97.91%**	0,0008	**1.02%**	0,0005
D_3	98.21%	0.89%	**98.72%**	0,0005	**0.63%**	0,0005
D_4	**98.53%**	**0.73%**	97.27%	0,0021	1.35%	0,0015
D_5	**98.49%**	**0.75%**	97.29%	0,002	1.36%	0,0014
D_6	98.48%	0.77%	**99.14%**	0,0003	**0.44%**	0,0006

Já para os resultados obtidos pelas medidas de avaliação *tc* e *ta*, ilustrados pela Tabela 7.11 mostram que as diferenças são estatisticamente significantes em alguns usuários. Para a medida *tc*, apenas os bancos de dados D_4 e D_5 a hipótese nula (Caso 1) é rejeitada. Para estes mesmos bancos de dados (D_4 e D_5) a hipótese nula do Caso 2 (medida de avaliação *ta*) é rejeitada.

Os resultados de significância estatísticas obtidos pela comparação do algoritmo FuzzyPrefMiner no cenário *fuzzy* com o algoritmo CPrefMiner, apresenta resultados semelhantes aos obtidos pela comparação com o algoritmo no cenário *crisp*. Portanto, este grupo de experimento ilustra o teste de significância estatística apenas do FuzzyPrefMiner no cenário *crisp* em comparação com o CPrefMiner. As Tabelas 7.12 e 7.13 destacam um resumo da comparação do FuzzyPrefMiner (*Crisp* e *Fuzzy*) com o CPrefMiner.

Tabela 7.12: CPrefMiner × FuzzyPrefMiner (*Crisp* e *Fuzzy*) *acc*, *revoc* e *prec*

BDP-F	CPrefMiner			FuzzyPrefMiner: *Crisp*			FuzzyPrefMiner: *Fuzzy*		
	acc	*revoc*	*prec*	*acc*	*revoc*	*prec*	*acc*	*revoc*	*prec*
D_1	81.24%	79.45%	82.34%	**87.41%**	**85.64%**	**88.80%**	81.60%	79.93%	82.77%
D_2	80.48%	79.46%	81.18%	**88.42%**	**87.17%**	**89.45%**	85.90%	84.73%	86.96%
D_3	83.85%	83.32%	84.32%	**90.09%**	**89.21%**	**90.83%**	88.33%	87.41%	89.13%
D_4	81.79%	80.44%	82.71%	**89.96%**	**89.23%**	**90.56%**	88.16%	87.44%	88.76%
D_5	81.36%	80%	82.25%	**89.25%**	**88.50%**	**89.85%**	86.97%	86.18%	87.65%
D_6	81.29%	80.86%	81.56%	**89.48%**	**88.70%**	**90.07%**	86.96%	86.18%	87.58%

Como previsto, os resultados do algoritmo FuzzyPrefMiner supera os resultados obtidos pelo CPrefMiner aplicados no mesmo cenário de mineração de preferências. Este grupo de teste valida a hipótese original deste trabalho de dissertação, pois, mostra que a

Tabela 7.13: CPrefMiner × FuzzyPrefMiner: (*Crisp* e *Fuzzy*) *tc* e *ta*

BDP-F	CPrefMiner		FuzzyPrefMiner: *Crisp*		FuzzyPrefMiner: *Fuzzy*	
	tc	*ta*	*tc*	*ta*	*tc*	*ta*
D_1	95.50%	1.78%	96.45%	1.76%	**96.47%**	**1.67%**
D_2	**97.91%**	**1.02%**	97.46%	1.25%	97.43%	1.17%
D_3	**98.72%**	**0.63%**	98.21%	0.89%	98.07%	0.92%
D_4	97.26%	1.35%	**98.53%**	0.73%	98.51%	**0.72%**
D_5	97.29%	1.36%	**98.49%**	**0.75%**	98.31%	0.79%
D_6	**99.14%**	**0.43%**	98.48%	0.77%	98.40%	0.77%

informação adicional do grau de preferência na entrada dos dados influencia o desempenho do algoritmo de mineração de preferências.

Com relação ao tempo de execução dos algoritmos a Tabela 7.14 mostra os resultados de tempo de execução dos dois algoritmos (CPrefMiner e FuzzyPrefMiner) tanto para criação do modelo de preferência, quanto para o uso do modelo.

A etapa de criação do modelo de preferência do algoritmo CPrefMiner, corresponde ao tempo que o algoritmo leva para construir uma RPB a partir de um conjunto de dados *crisp*. Enquanto o tempo de criação do modelo de preferência do algoritmo FuzzyPrefMiner, corresponde a soma dos tempos de criação das partições do BDP-F pelas faixas de preferências (neste grupo de teste 8 faixas de preferências são consideradas), o tempo de treinamento do classificador $\mathcal{M}$ e o tempo de criação das 8 RPBs a partir das partições realizadas pelas faixas de preferências. Como esperado, o tempo de criação do modelo de preferência do algoritmo CPrefMiner é menor que o tempo gasto pelo algoritmo FuzzyPrefMiner.

Tabela 7.14: Tempo de Execução - Criação e Uso do Modelo de Preferência

BDP-F	**Criação do Modelo**	
	CPrefMiner	FuzzyPrefMiner
D_1	1 seg 25 mseg	3 seg 93 mseg
D_2	2 seg 933 mseg	4 seg 667 mseg
D_3	5 seg 431 mseg	8 seg 427 mseg
D_4	8 seg 914 mseg	14 seg 117 mseg
D_5	13 seg 3 mseg	20 seg 241 mseg
D_6	15 seg 442 mseg	26 seg 835 mseg

Uso do Modelo	
CPrefMiner	FuzzyPrefMiner
35 mseg	50 mseg

Quando se trata do tempo de uso do modelo, o algoritmo CPrefMiner também consegue valores melhores. Neste tempo de execução, são consideradas a comparação de 1000 pares de tuplas nos dois algoritmos. Este aumento no tempo de uso do FuzzyPrefMiner já era esperado, uma vez que, antes de realizar a comparação de tuplas pela RBP, é preciso aplicar o classificador treinado e selecionar a RPB correspondente. No entanto, pode-se notar que o aumento não é tão grande quando se comparado ao tempo de uso do modelo pelo algoritmo CPrefMiner.

7.3 Validação dos Métodos de Reparação de Inconsistência

Esta seção é voltada para avaliar as inconsistências das relações de preferências *fuzzy* inferidas pelo algoritmo FuzzyPrefMiner e avaliar o qualidade do modelo após a aplicação dos métodos de reparação de inconsistência.

7.3.1 Avaliação da Inconsistência

Antes de aplicar os métodos de reparação de inconsistência, é importante destacar que as matrizes inferidas pelo FuzzyPrefMiner não são consistentes com relação a propriedade de transitividade fraca.

Para avaliar o quanto as matrizes inferidas pelo método são inconsistentes, o Algoritmo 5 que contabiliza a quantidade de ciclos de tamanho 3 descrito no Capítulo 6 foi aplicado. A Tabela 7.15 apresenta a média de ciclos das 30 iterações do *cross-validation* para as matrizes inferidas pelo algoritmo FuzzyPrefMiner.

Tabela 7.15: Média de Ciclos por BDP-F

BDP-F	**Tamanho Matriz**	**Média de Ciclos (mc)**
D_1	300×300	22.33
D_2	610×610	192.07
D_3	996×996	786.17
D_4	1214×1214	1519.47
D_5	1454×1454	2335.87
D_6	1688×1688	3734.9

Cada ciclo representa uma contradição a propriedade de transitividade fraca. Nota-se, que o método produz relações de preferências *fuzzy* inconsistentes. Portanto, os grupos de testes apresentados nas próximas subseções visa reparar essas inconsistências e com isso avaliar novamente a qualidade do modelo de preferência.

7.3.2 Resultados da Técnica *Range Voting*

Neste grupo de teste, o objetivo é analisar a qualidade do modelo de preferência quando a técnica de Range Voting é aplicada para reparar a inconsistência das matrizes inferidas pelo FuzzyPrefMiner.

As técnicas de *Range Voting* Não Incremental e Incremental descritas na Seção 6.2 do Capítulo 6 foram aplicadas e os resultados descritos na Tabela 7.16 foram obtidos.

Nota-se que em ambas as versões do *Range Voting* os valores obtidos pelas medidas *acc*, *revoc* e *prec* são iguais. Isso acontece pelo fato que a Técnica de *Range Voting* consegue comparar todos os pares de tuplas, ou seja, o módulo aleatório não é acionado

Tabela 7.16: Resultados *Range Voting*

BDP-F	Range Voting Não Incremental				Range Voting Incremental			
	acc	*revoc*	*prec*	*mc*	*acc*	*revoc*	*prec*	*mc*
D_1	**94.57%**	**94.57%**	**94.57%**	0.47	94.44%	94.44%	94.44%	**0**
D_2	96.97%	96.97%	96.97%	7.13	**97.55%**	**97.55%**	**97.55%**	**0**
D_3	97.32%	97.32%	97.32%	57.13	**98.41%**	**98.41%**	**98.41%**	**0**
D_4	97.05%	97.05%	97.05%	121.83	**98.02%**	**98.02%**	**98.02%**	**0**
D_5	95.93%	95.93%	95.93%	337.6	**97.77%**	**97.77%**	**97.77%**	**0**
D_6	96.75%	96.75%	96.75%	293.37	**97.64%**	**97.64%**	**97.64%**	**0**

Tabela 7.17: Taxas de Comparabilidade e Aleatoriedade

BDP-F	Range Voting Não Incremental		Range Voting Incremental	
	tc	*ta*	*tc*	*ta*
D_1	**100%**	**0%**	**100%**	**0%**
D_2	**100%**	**0%**	**100%**	**0%**
D_3	**100%**	**0%**	**100%**	**0%**
D_4	**100%**	**0%**	**100%**	**0%**
D_5	**100%**	**0%**	**100%**	**0%**
D_6	**100%**	**0%**	**100%**	**0%**

em nenhum momento. Logo, tanto para a versão não incremental quanto para a versão incremental os mesmos valores de taxas de comparabilidade e aleatoriedade são obtidos. Como pode ser visto nos resultados na Tabela 7.17 são obtidos valores de 100% de taxa de comparabilidade e consequentemente 0% de aleatoriedade.

É notável na Tabela 7.16, que a versão Incremental da técnica consegue melhores resultados que a versão não incremental, como previsto. Somente para o primeiro BDP-F D_1 que a versão não incremental é um pouco superior a versão incremental.

Tabela 7.18: Resumo: Média de Ciclos das Matrizes de Preferências

BDP-F	**FuzzyPrefMiner**	**RV não Incremental**	**RV Incremental**
D_1	22.33	0.47	**0**
D_2	192.07	7.13	**0**
D_3	786.17	57.13	**0**
D_4	1519.47	121.83	**0**
D_5	2335.87	337.6	**0**
D_6	3734.9	293.37	**0**

Quando se trata da média de ciclos encontrados nas matrizes de preferências, nota-se, que o *Range Voting* Não Incremental não consegue eliminar todos os ciclos da matriz, ao contrário da versão incremental que elimina todos os ciclos. Mesmo não conseguindo eliminar todos os ciclos, o *Range Voting* Não Incremental consegue diminuir consideravelmente a média de ciclos da matriz de preferência original inferida pelo FuzzyPrefMiner, como pode ser analisado no resumo ilustrado pela Tabela 7.18.

Com relação ao tempo de execução, o Range Voting Não Incremental executa em menos tempo que a versão Incremental, como pode ser visto na Tabela 7.19.

Tabela 7.19: Tempo de Execução da Técnica *Range Voting*

BDP-F	*RV Não Incremental*	*RV Incremental*
D_1	**128 mseg**	1 seg 975 mseg
D_2	**1 seg 87 mseg**	20 seg 59 mseg
D_3	**3 seg 965 mseg**	1 min 22 seg
D_4	**8 seg 606 mseg**	3 min 56 seg
D_5	**17 seg 63 mseg**	8 min 5 seg
D_6	**27 seg 493 mseg**	13 min38 seg

7.3.3 Comparação com Método BTF (Xu *et al.*, 2013)

Este grupo de teste tem como objetivo, validar o método de reparação de inconsistência proposto nesta dissertação baseado na técnica de Range Voting. Para realizar essa validação, é necessário comparar essa técnica com um método de reparação de inconsistência de relações de preferências *fuzzy* existente na literatura.

O trabalho escolhido como *baseline* de comparação para este grupo de teste é o método proposto por Xu *et al.* (2013), descrito na Seção 6.1 do Capítulo 6. Este método é recursivo e converge quando o número de ciclos de tamanho 3 de uma matriz de preferência obtém valor 0 (zero). Além disso, este método converge apenas em matrizes cheias, e devido a estratificação utilizada no pré-processamento dos dados as matrizes de preferências inferidas pelo FuzzyPrefMiner são bastante esparsas. Portanto, é necessário preencher os valores em falta da matriz de preferência *fuzzy* inferida antes da aplicação do método de reparação de inconsistência.

Essa esparsidade acontece devido ao protocolo *Weak Protocol* utilizado na estratificação dos dados. Por exemplo, suponha um BDP-F D_i e uma partição $D_{i_k} \in D_i$ utilizada para teste do modelo ilustrada pela Figura 7.1(a). A matriz de preferência obtida deste conjunto ilustrada pela Figura 7.1 (b) é esparsa, pois não contém todos os pares de tuplas possíveis no conjunto de teste.

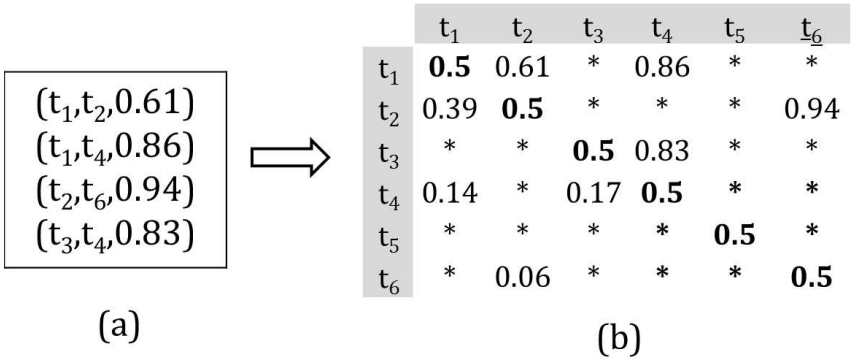


Figura 7.1: (a) Partição D_{i_k} de Teste e (b) Matriz de Preferência

A partir dessa matriz, é possível calcular a porcentagem de preenchimento da matriz pela fórmula $p = \frac{ep*100}{te}$, onde, ep é o número de elementos preenchidos na matriz e te é o total de elementos que a matriz comporta. Neste exemplo, $ep = 14$ e $te = 6 \times 6 = 36$, logo,

$p=\frac{14*100}{36} = 38.89\%$. Este valor p representa a porcentagem de preenchimento da matriz de preferência pelos pares de tuplas de uma partição D_{i_k} usada para teste do modelo.

Para cada banco de dados D_i escolhido para os experimentos, foi calculada a porcentagem de preenchimento da matriz que representa a partição D_{i_k} escolhida para teste do modelo. Os resultados ilustrados pela Tabela 7.20, mostram que em todos os bancos de dados, uma porcentagem (p) muito pequena da matriz é preenchida, em torno de aproximadamente 3%, o que resulta em alta esparsidade dos dados.

Tabela 7.20: Porcentagem de Preenchimento das matrizes

BDP-F	**Tamanho Matriz**	p
D_1	300×300	2.89%
D_2	610×610	2.74%
D_3	996×996	2.36%
D_4	1214×1214	2.63%
D_5	1454×1454	2.71%
D_6	1688×1688	2.65%

Para tratar a esparsidade dessas matrizes, o FuzzyPrefMiner é aplicado em todos os pares de tuplas possíveis da matriz de preferência, mesmo que estes não façam parte do banco de dados de teste. Dessa forma, é possível obter uma matriz completamente cheia. No entanto, o número de ciclos aumenta consideravelmente como ilustra a Tabela 7.21.

Tabela 7.21: Média de Ciclos por BDP-F

BDP-F	**Tamanho Matriz**	mc_e	mc_c
D_1	300×300	22.33	18.540,30
D_2	610×610	192.07	105.849,30
D_3	996×996	786.17	311.870,12
D_4	1214×1214	1519.47	657.682,90
D_5	1454×1454	2335.87	1.249.524,38
D_6	1688×1688	3734.9	2.134.576,43

É notável, um grande aumento no número de ciclos de uma matriz completamente cheia. Portanto, é esperado que o tempo de execução do método BTF também seja muito alto, pois ele aumenta de acordo com o número de vezes que é aplicado para obter uma matriz de preferência consistente, isto é, o mesmo é aplicado até que o número de ciclos $C = 0$.

O objetivo deste grupo de teste é comparar a técnica de *Range Voting* Incremental com o Método BTF, pois ambos conseguem eliminar todos os ciclos de tamanho 3 da matriz. Os resultados obtidos por esse grupo de teste pode ser visualizado nas Tabelas 7.22 e 7.23.

Com relação as medidas de avaliação acurácia e precisão, nota-se, que o método BTF alcança valores superiores ao *Range Voting* Incremental. Enquanto, para a medida de

Tabela 7.22: Comparação: RV Incremental × Método BTF (*acc, revoc* e *prec*)

BDP-F	RV Incremental				Método BTF			
	acc	*revoc*	*prec*	*mc*	*acc*	*revoc*	*prec*	*mc*
D_1	94.44%	94.44%	94.44%	**0**	**97.99%**	**94.63%**	**97.94%**	**0**
D_2	97.55%	**97.55%**	97.55%	**0**	**99.18%**	96.87%	**99.16%**	**0**
D_3	98.41%	**98.41%**	98.41%	**0**	**99.54%**	97.71%	**99.53%**	**0**
D_4	98.02%	**98.02%**	98.02%	**0**	**99.09%**	97.67%	**99.08%**	**0**
D_5	97.77%	**97.77%**	97.77%	**0**	**99.19%**	97.61%	**98.18%**	**0**
D_6	97.64%	**97.64%**	97.64%	**0**	**98.74%**	97.22%	**98.73%**	**0**

Tabela 7.23: Comparação: RV Incremental × Método BTF (*tc* e *ta*)

BDP-F	RV Incremental		Método BTF	
	tc	*ta*	*tc*	*ta*
D_1	**100%**	**0%**	96.61%	3.36%
D_2	**100%**	**0%**	97.68%	2.32%
D_3	**100%**	**0%**	98.17%	1.83%
D_4	**100%**	**0%**	98.58%	1.42%
D_5	**100%**	**0%**	98.41%	1.58%
D_6	**100%**	**0%**	98.47%	1.53%

revocação, os valores obtidos pelo Range Voting Incremental são superiores na maiores dos BDP-Fs. É notável também, que ao contrário do *Range Voting* o Método BTF não obtém os mesmos valores para as medidas de acurácia, revocação e precisão. Isso acontece porque o método BTF não obtém 100% e taxa de comparabilidade e 0% de taxa de aleatoriedade, como mostra a Tabela 7.23.

O Método BTF não alcança 100% de comparabilidade pelo fato que o mesmo não utiliza os valores das probabilidades p_1 e p_2 inferidas pelo FuzzyPrefMiner para reparar a inconsistência, ao contrário do *Range Voting*. Isto é, o Método BTF trabalha com os mesmos graus de preferências inferidos pelo FuzzyPrefMiner, modificando apenas o sentido da intensidade de preferência quando uma posição da matriz é considerada entrada de inconsistência. Já o *Range Voting* atua a partir das inferências da ordenação de preferência entre duas tuplas para encontrar o novo grau de preferência de um par de tuplas.

É notável que o Método BTF consegue melhores resultados em algumas medidas de avaliações (acurácia e precisão), no entanto, quando se trata da comparabilidade e aleatoriedade o *Range Voting* é superior em todos os banco de dados.

Ainda comparando os dois métodos, o tempo de execução para efetuar a reparação da inconsistência das matrizes de preferências foi avaliado e como esperado, o tempo de execução do método BTF é superior ao tempo de execução do *Range Voting* Incremental (veja Tabela 7.24).

O tempo de execução do Método BTF é muito alto, pois o mesmo cresce de acordo com o número de vezes que é aplicado em uma matriz de preferência para se obter o número de ciclos $C = 0$. Como pode-se notar na Tabela 7.21, o número de ciclos aumenta

Tabela 7.24: Tempo de Execução da Técnica *Range Voting* × Método BTF

BDP-F	*RV Incremental*	*Método BTF*
D_1	**1 seg 975 mseg**	12 seg 759 mseg
D_2	**20 seg 59 mseg**	3 min 98 seg
D_3	**1 min 22 seg**	29 min 71 seg
D_4	**3 min 56 seg**	2 hrs 3 min 34 seg
D_5	**8 min 5 seg**	5 hrs 56 min 26 seg
D_6	**13 min38 seg**	1 d 11 hrs 2min 20 seg

consideravelmente quando o FuzzyPrefMiner é aplicado em todos os pares de tuplas para obtenção da matriz cheia, portanto, eliminar todos esses ciclos pelo método BTF torna-se um processo custoso.

7.4 Considerações do Capítulo

Neste capítulo, foi apresentado uma série de experimentos em dados reais para validação do algoritmo *FuzzyPrefMiner* e também, dos métodos de reparação propostos nesta dissertação baseados na técnica de Range Voting (de Amo *et al.*, 2014).

O algoritmo FuzzyPrefMiner se mostrou superior ao algoritmo CPrefMiner (de Amo *et al.*, 2012a), mesmo quando validado no cenário *Fuzzy*. Em relação ao tempo de execução o FuzzyPrefMiner perde quando comparado ao CPrefMiner, uma vez que, o processo de construção do modelo de preferência pelo algoritmo FuzzyPrefMiner possui diversas etapas, tais como, divisão do BDP-F por k faixas de preferências, treinamento do classificador e construção das k redes de preferências *bayesianas*.

Com relação aos métodos de reparação de inconsistência das relações de preferências *fuzzy*, a técnica *Range Voting* Incremental conseguiu melhor desempenho quando comparado ao *Range Voting* Não Incremental, pois, o mesmo consegue retirar todos os ciclos de tamanho 3 da matriz de preferência com base em uma técnica de votação.

Quando comparado, o *Range Voting* Incremental com o Método BTF (Xu *et al.*, 2013), nota-se que o método BTF consegue resultados um pouco melhores que o *Range Voting* em termos de acurácia e precisão. No entanto, para as medidas de revocação, taxa de comparabilidade e taxa de aleatoriedade o *Range Voting* Incremental consegue resultados superiores.

Além disso, o tempo de execução do *Range Voting* é extremamente inferior ao tempo de execução do Método BTF, uma vez que, o método BTF é um algoritmo recursivo que termina quando o número de ciclos de tamanho 3 obtém valor 0. Este tempo de execução se tornou alto, devido ao grande número de ciclos de tamanho 3 das matrizes de preferência obtidas após o tratamento da esparsidade, pois, além dos ciclos reais das matrizes esparsas, existe o acréscimo dos ciclos encontrados após o preenchimento das matrizes.

Capítulo

8

Conclusão e Trabalhos Futuros

Neste trabalho de dissertação, é apresentado primeiramente um estudo sobre a influência do *grau de preferência* no desempenho de técnicas de mineração de preferências, especificadas pelo modelo de preferência contextual. Um formalismo para especificar um Modelo de Preferência Contextual *Fuzzy* constituído por um classificador e um conjunto de Redes de Preferências Bayesianas é proposto. Este modelo generaliza o Modelo de Preferência Contextual tratado em pesquisas anteriores no cenário *crisp*.

Foi realizada uma adaptação do algoritmo CPrefMiner (de Amo *et al.*, 2012a), desenvolvido para minerar modelos de preferências contextuais *crisp*, para trabalhar em tarefas de mineração de preferências contextuais *fuzzy*. Essa nova versão do algoritmo CPrefMiner nomeada FuzzyPrefMiner trabalha com dados de entrada mais ricos em informações (o grau de preferência) e também produz resultados mais refinados.

Um extenso conjunto de experimentos executados em dados reais mostram que a hipótese original está correta, ou seja, dados de preferências *fuzzy* desempenham um papel importante na concepção de métodos eficientes para a mineração de preferências baseados em comparação de pares de tuplas. Os experimentos mostram também, que o método é capaz de produzir predições mais refinados (o grau de preferência entre duas tuplas) sem perda de precisão, no que diz respeito a um método desenvolvido para retornar predições *crisp* (a preferência entre duas tuplas).

Neste trabalho de mestrado, também é apresentado o desenvolvimento de métodos de reparação de inconsistência de relações de preferências *fuzzy* inferidas pelo algoritmo FuzzyPrefMiner, quando executado cenário *fuzzy*. Uma RPB introduzida em trabalhos anteriores (de Amo *et al.*, 2012a),(de Amo *et al.*, 2013) é capaz de inferir uma *ordem parcial estrita* sobre um conjunto de pares de tuplas, no entanto, quando se trata de uma relação

de preferência *fuzzy* inferida pelo FuzzyPrefMiner, esta não verifica algumas propriedades de relações de preferências *fuzzy*, tais como, transitividade fraca e transitividade aditiva (Herrera-Viedma *et al.*, 2004).

Para solucionar este problema, são apresentados dois métodos de reparação de inconsistência, baseados em técnicas de sistemas de votação (de Amo *et al.*, 2014). O primeiro método denominado *Range Voting* Não Incremental apresentou resultados satisfatórios com relação a precisão na predição dos graus de preferências, no entanto não conseguiu reparar toda a inconsistência de uma relação de preferência *fuzzy*. O segundo método de reparação de inconsistência apresentado nesta dissertação, é um refinamento da primeira versão, nomeado *Range Voting* Incremental. Este segundo método apresentou melhores resultados que a versão não incremental, pois além de aumentar a precisão do modelo de preferência, o mesmo conseguiu eliminar todas as inconsistências da matriz que representa uma relação de preferência *fuzzy*.

Para validar o método de reparação de inconsistência baseado em técnicas de votação, o mesmo é comparado com um método existente na literatura (Xu *et al.*, 2013). Os experimentos foram realizados sobre as relações de preferências *fuzzy* inferidas pelo algoritmo FuzzyPrefMiner, e mostram que o Método BTF é superior em termos de acurácia e precisão. No entanto, tem termos de revocação, comparabilidade e aleatoriedade o método *Range Voting* Incremental consegue resultados superiores.

Finalmente, pode-se concluir que os dados de preferências *fuzzy* melhoram o desempenho de algoritmo de mineração de preferências que possuem o modelo de preferência contextual subjacente. Além disso, é notável que a técnica de votação *Range Voting* usada para reparar inconsistência de relações de preferências *fuzzy* consegue resultados bastante satisfatórios.

8.1 Publicações

Durante o desenvolvimento deste trabalho de mestrado foi publicado um artigo entitulado *"Mineração de Preferências Contextuais Fuzzy"* na *16th International Conference on Data Warehousing and Knowledge Discovery - DaWaK 2014*.

Pretende-se ainda a curto prazo realizar a submissão de um artigo para o *2nd Symposium on Knowledge Discovery, Mining and Learning* (KDMILE 2004) com submissão dia 01 de Julho de 2014. Outro artigo para o XXIX Simpósio Brasileiro de Bancos de Dados (SBBD 2014) com previsão de submissão em meados de Junho e um terceiro artigo a ser submetido para a *IEEE International Conference on Tools with Artificial Intelligence* (ICTAI 2014), com previsão de submissão em 30 de Junho de 2014.

A longo prazo pretende-se submeter todo o trabalho descrito nesta dissertação de mestrado em um *Journal* (a ser definido).

8.2 Trabalhos Futuros

Durante o desenvolvimento da pesquisa, foram levantadas várias questões possíveis para otimização do algoritmo FuzzyPrefMiner e dos métodos de reparação de inconsistência.

1. Utilizar outras funções de transformação para obter as relações de preferências *fuzzy*.

   O uso de outras funções de transformação para transformar notas de tuplas em relações de preferências *fuzzy* é interessante para avaliar o desempenho do algoritmo FuzzyPrefMiner. Duas possibilidades são estudadas para variar essa função:

   - Variar o valor "n" da função $f_n(\frac{u_i}{u_j}) = \frac{u_i^n}{u_i^n + u_j^n}$ proposta por Tanino (1984) e Chiclana *et al.* (1998), para avaliar se outros valores para n conseguem melhores resultados.
   - Utilizar a função proposta por Ma *et al.* (2006). Esta função é diferente da família de funções baseadas na escala de razão utilizadas neste trabalho, pois, com seu uso é garantido a propriedade de transitividade aditiva. Portanto, um trabalho futuro interessante, seria verificar o comportamento deste tipo de função de transformação no algoritmo FuzzyPrefMiner e consequentemente analisar a consistência das relações de preferências *fuzzy* inferidas pelo mesmo.

2. Realizar um estudo aprofundado sobre a propriedade de transitividade aditiva.

   Com este estudo objetiva-se propor novos métodos de reparação de inconsistência de matrizes de preferências *fuzzy*.

3. Realizar um estudo para encontrar novas formas de prever o grau de preferência de um par de tuplas.

   Com base na inferência do algoritmo FuzzyPrefMiner, é interessante realizar um estudo para encontrar outras formas de inferir o grau de preferência de forma que, o método consiga o mesmo ou melhor desempenho do FuzzyPrefMiner atual. Tal estudo deverá ser concentrado na propriedade de transitividade aditiva, pois, trabalhos na literatura (Xu *et al.*, 2013),(Herrera-Viedma *et al.*, 2007), (Liu *et al.*, 2012) mostram que, preencher matrizes com valores baseados nessa propriedade garantem a consistência das relações de preferência *fuzzy*.

4. Realizar testes em dados sintéticos

   A realização de testes em dados sintéticos para avaliar os algoritmos quando aplicados em dados completamente diferentes dos dados utilizados até momento. É necessário propor e implementar um gerador de dados sintéticos baseado em matrizes de preferências, para não ter o problema de matrizes esparsas como acontece

nos dados reais e assim avaliar melhor a qualidade dos métodos de reparação de inconsistência.

5. Testar o algoritmo FuzzyPrefMiner em outros cenários de preferências

   Realizar testes com o algoritmo FuzzyPrefMiner em outros cenários que não sejam avaliação de filmes, para analisar quão bem este algoritmo se comporta em outros cenários.

6. Acoplar o algoritmo FuzzyPrefMiner em um Sistema de Recomendação

   Testar o algoritmo FuzzyPrefMiner acoplado em Sistemas de Recomendação que utilizam algoritmos de mineração de preferências e avaliar o impacto do algoritmo neste ambiente.

Referências

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., e Poole, D. (2004). CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. páginas 135–191.

Burges, C. J. C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., e Hullender, G. N. (2005). Learning to rank using gradient descent. In *ICML*, páginas 89–96.

Chiclana, F., Herrera, F., e Herrera-Viedma, E. (1998). Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets and Systems*, 97(1):33–48.

Chiclana, F., Herrera, F., e Herrera-Viedma, E. (2001). Integrating multiplicative preference relations in a multipurpose decision-making model based on fuzzy preference relations. *Fuzzy Sets and Systems*, 122(2):277–291.

Cohen, W. W., Schapire, R. E., e Singer, Y. (1999). Learning to Order Things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270.

Crammer, K. e Singer, Y. (2001). Pranking with Ranking. In *NIPS*, páginas 641–647.

de Amo, S., Bueno, M. L. P., Alves, G., e Silva, N. F. (2012a). CPrefMiner: An Algorithm for Mining User Contextual Preferences based on Bayesian Networks. *24th IEEE International Conference on Tools with Artificial Intelligence*.

de Amo, S., Diallo, M. S., Diop, C., Giacometti, A., Li, H., e Soulet, A. (2012b). Mining Contextual Preference Rules for Building User Profiles. In *Data Warehousing and Knowledge Discovery*, volume 7448, páginas 229–242. Springer Berlin Heidelberg.

de Amo, S., Bueno, M. L. P., Alves, G., e da Silva, N. F. F. (2013). Mining User Contextual Preferences. *JIDM*, 4(1):37–46.

de Amo, S., Diallo, M. S., Giacometti, A., Li, H., e Soulet, A. (2014). Contextual Preference Mining for User Profile Construction. *To appear in Information Systems (Elsevier)*.

de Sá, C. R., Soares, C., Jorge, A. M., Azevedo, P. J., e da Costa, J. P. (2011). Mining Association Rules for Label Ranking. In *PAKDD (2)*, páginas 432–443.

Eckhardt, A. e Vojtás, P. (2009). How to Learn Fuzzy User Preferences with Variable Objectives. In *IFSA/EUSFLAT Conf.*, páginas 938–943.

Freund, Y., Iyer, R., Schapire, R. E., e Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969.

Fürnkranz, J. e Hüllermeier, E. (2011). *Preference Learning*.

Herrera-Viedma, E., Herrera, F., Chiclana, F., e Luque, M. (2004). Some issues on consistency of fuzzy preference relations. *European Journal of Operational Research*, 154(1):98–109.

Herrera-Viedma, E., Chiclana, F., Herrera, F., e Alonso, S. (2007). Group Decision-Making Model With Incomplete Fuzzy Preference Relations Based on Additive Consistency. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):176–189.

Hüllermeier, E., Fürnkranz, J., Cheng, W., e Brinker, K. (2008). Label ranking by learning pairwise preferences. páginas 1897–1916.

Holland, S., Ester, M., e Kießling, W. (2003). Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications. In *Knowledge Discovery in Databases: PKDD 2003*, volume 2838, páginas 204–216. Springer Berlin Heidelberg.

Jansen, F. V. e Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. 2nd ed. edition.

Jiang, B., Pei, J., Lin, X., Cheung, D. W., e Han, J. (2008). Mining preferences from superior and inferior examples. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 390–398. ACM.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *KDD*, páginas 133–142.

Koriche, F. e Zanuttini, B. (2010). Learning conditional preference networks. páginas 685–703.

Larranaga, P., Poza, M., Yurramendi, Y., Murga, R., e Kuijpers, C. (1996). Structure learning of Bayesian networks by genetic algorithms: a performance analysis of control parameters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):912–926.

Liu, X., Pan, Y., Xu, Y., e Yu, S. (2012). Least square completion and inconsistency repair methods for additively consistent fuzzy preference relations. *Fuzzy Sets and Systems*, 198(0):1 – 19.

Lops, P., de Gemmis, M., e Semeraro, G. (2011). Content-based Recommender Systems:: State of the Art and Trends. *Recommender Systems Handbook*, páginas 73–105.

Ma, J., Fan, Z.-P., Jiang, Y.-P., Mao, J.-Y., e Ma, L. (2006). A method for repairing the inconsistency of fuzzy preference relations. *Fuzzy Sets and Systems*, 157(1):20–33.

Stefanidis, K., Koutrika, G., e Pitoura, E. (2011). A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst.*, 36:19:1–19:45.

Tanino, T. (1984). Fuzzy preference orderings in group decision making. *Fuzzy Sets and Systems*, 12(2):117–131.

Urdan, T. (2010). *Statistics in Plain English, Third Edition*. Taylor & Francis.

Wilson, N. (2004). Extending CP-Nets with Stronger Conditional Preference Statements. In *AAAI*, páginas 735–741.

Xu, Y., Patnayakuni, R., e Wang, H. (2013). The ordinal consistency of a fuzzy preference relation. *Information Sciences*, 224(0):152–164.

Zenebe, A., Zhou, L., e Norcio, A. F. (2010). User preferences discovery using fuzzy models. *Fuzzy Sets and Systems*, 161(23):3044–3063.

Zhang, G., Dong, Y., e Xu, Y. (2012). Linear optimization modeling of consistency issues in group decision making based on fuzzy preference relations. *Expert Systems with Applications*, 39(3):2415 – 2420.