Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial

Biclusterização na Análise de Dados Incertos

Fabrício Olivetti de França

Orientador: Fernando J. Von Zuben

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para a obtenção do título de Doutor em Engenharia Elétrica.

Área de concentração: Engenharia de Computação.

Campinas - SP - Brasil

Novembro de 2010

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

França, Fabrício Olivetti de
F844b Biclusterização na análise de dados incertos / Fabrício Olivetti de França. --Campinas, SP: [s.n.], 2010.

Orientador: Fernando José Von Zuben.
Tese de Doutorado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Aprendizado de computador. 2. Dados faltantes (Estatística). 3. Cluster. 4. Mineração de dados (Computação). 5. Algoritmos evolutivos. I. Von Zuben, Fernando José. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Biclustering on uncertain data analysis
Palavras-chave em Inglês: Computer training, Missing data (Statistics), Cluster, Data mining (Computer), Evolutionary algorithms
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: André Luís Vasconselos Coelho, Fernando Buarque de Lima Neto, Christiano Lyra Filho, Romis Ribeiro de Faissol Attux
Data da defesa: 29/11/2010
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Fabrício Olivetti De França

Data da Defesa: 29 de novembro de 2010

Título da Tese: "Biclusterização na Análise de Dados Incertos"

Prof. Dr. Fernando José Von Zuben (Presidente): Fernando José Von Zuben

Prof. Dr. André Luís Vasconcelos Coelho:

Prof. Dr. Fernando Buarque de Lima Neto:

Prof. Dr. Christiano Lyra Filho:

Prof. Dr. Romis Ribeiro de Faissol Attux:

Resumo

O processo de aquisição de dados está sujeito a muitas fontes de incerteza e inconsistência. Essas incertezas podem fazer com que os dados se tornem ruidosos ou impedir a aquisição dos mesmos, gerando o problema de dados faltantes. A maioria das ferramentas utilizadas para tratar tais problemas age de forma global em relação às informações da base de dados e ignora o efeito que o ruído pode ter na análise desses. Esta tese tem como objetivo explorar as propriedades do processo de biclusterização, que faz uma análise local dos dados, criando múltiplos modelos de imputação de dados que buscam minimizar o erro de predição dos valores faltantes na base de dados. Primeiramente, é proposto um novo algoritmo de biclusterização com um melhor desempenho que outras abordagens utilizadas atualmente, enfatizando a capacidade dos biclusters em gerar modelos com ruído reduzido. Em seguida, é proposta uma formulação de otimização quadrática para, utilizando os modelos locais gerados pelo bicluster, imputar os valores faltantes na base de dados. Os resultados obtidos indicam que a utilização da biclusterização ajuda a reduzir o erro de predição da imputação, além de fornecer condições favoráveis a uma análise *a posteriori* das informações contidas nos dados.

Palavras-chave: biclusterização, imputação de dados, aprendizado de máquina, dados faltantes, dados ruidosos, meta-heurísticas.

Abstract

The data acquisition process is subject to many inconsistencies and uncertainties. These uncertainties may produce noisy data or even provoke the absence of some of them, thus leading to the missing data problem. Most procedures used to deal with such problem act in a global manner, relatively to the dataset, and ignore the noise effect on such analysis. The objective of this thesis is to explore the properties of the so called biclustering method, which performs a local data analysis, creating several imputation models for the dataset in order to minimize the prediction error estimating missing values of the dataset. First, it is proposed a new biclustering algorithm with a better performance than the one produced by other traditional approaches, with emphasis on the noise reduction capability of the models generated by the biclusters. Next, it is proposed the formulation of a quadratic optimization problem to impute the missing data by means of the local models engendered by a set of biclusters. The obtained results show that the use of biclustering helps to reduce the prediction error of data imputation, besides providing some interesting conditions for an *a posteriori* analysis of the dataset.

Keywords: biclustering, data imputation, machine learning, missing data, noisy data.

Agradecimentos

Ao meu orientador, Prof. Fernando J. Von Zuben, sou grato pela orientação.

Aos colegas de pós-graduação e à banca examinadora, pelas críticas e sugestões.

À minha família, pelo apoio durante esta jornada.

À Faculdade de Engenharia Elétrica e de Computação da Unicamp, por fornecer plenas condições para o desenvolvimento da pesquisa.

À CAPES, pelo suporte financeiro.

À minha família e amigos

Sumário

Lista de Figuras

Lista de Tabelas

Acrônimos

ACO - Ant Colony Optimization

AS - Ant Systems

BIC-aiNet - Artificial Immune Network for Biclustering

bicACO - Biclustering Ant Colony Optimization

CC - Algoritmo de Cheng & Church

GSA - Gravitational Search Algorithm

HD - Hot-Deck

IAT - Imputação baseado em outro Atributo

IVM - Imputação pelo Valor Médio

IWD - Intelligent Water Drops

k-VMP - k-Vizinhos Mais Próximos

MAR - Missing at Random

MCAR - Missing Completely at Random

MI - Múltipla Imputação

MMAS - Max-Min Ant System

MOM-aiNet - Multi-Objective Multipopulation Artificial Immune Network for Biclustering

NMAR - Not Missing at Random

NSGA-II - Non-Dominated Sorting Genetic Algorithm-II

PCV - Problema do Caixeiro Viajante

PSO - Particle Swarm Optimization

RD - Redução de Dados

RL - Regressão Linear

RQM - Resíduo Quadrático Médio

rSVD - Regulated Singular Value Decomposition

SDS - Stochastic Diffusion Search

SPC - Superparamagnetic Clustering

SwarmBcluster - Swarm Intelligence Biclustering

VMP - Vizinho mais próximo

Lista de Símbolos

A	-	Matriz de dados genérica
n	-	Número de linhas da matriz original de dados
m	-	Número de colunas da matriz original de dados
$A(I, J)$	-	Sub-matriz da matriz A contendo as linhas e colunas dos conjuntos I e J, respectivamente
a_{ij}	-	Elemento da i-ésima linha e j-ésima coluna da matriz A
a_{Ij}	-	Valor médio da j-ésima coluna da matriz A, levando em conta o sub-conjunto de linhas I
a_{iJ}	-	Valor médio da i-ésima linha da matriz A, levando em conta o sub-conjunto de colunas J
a_{IJ}	-	Média dos valores da matriz A, levando em conta os sub-conjuntos I e J de linhas e colunas, respectivamente
L	-	Conjunto de linhas da matriz A
C	-	Conjunto de colunas da matriz A
I	-	Sub-conjunto de linhas da matriz A
J	-	Sub-conjunto de colunas da matriz A
B	-	Conjunto de biclusters referentes à matriz A
B_l	-	l-ésimo bicluster definido por $B_l = A(I_l, J_l)$
k	-	Número de linhas de um bicluster
s	-	Número de colunas de um bicluster
$\tau_{i,j}$	-	Feromônio depositado na aresta (i, j)

Capítulo 1

Introdução

A tarefa de análise de dados pode ser interpretada como o estudo de técnicas para processamento de dados provenientes de diversas fontes, com o objetivo de evidenciar informações pertinentes para algum processo de tomada de decisão. Ela pode receber nomes distintos, dependendo da área de aplicação, como, por exemplo, *Mineração de Dados* (Han & Kamber, 2006) e *Análise Exploratória de Dados* (Hartwig & Dearing, 1979).

Com a massificação da aquisição de dados em vários setores, tornou-se possível dispor de bases de dados de grande porte, o que trouxe a necessidade da criação de técnicas especificamente voltadas para o seu tratamento, dentre as quais se destacam: técnicas capazes de extrair da base apenas os objetos e atributos relevantes para a análise a ser feita e técnicas de análise capazes de operar grande quantidade de dados sem perda de desempenho.

Um desafio para essas técnicas está na incerteza dos valores da base de dados a ser analisada. Nesta tese, o contexto de incerteza se refere à exatidão e qualidade dos valores obtidos. Essa incerteza pode ocorrer por diversos motivos, como imprecisão dos instrumentos de medição, incapacidade de obtenção de determinados valores e falha de processo. Dessa forma, duas situações podem ocorrer conjuntamente na base de dados: obtenção de valores ruidosos e ausência de valores na base.

Ambas as situações podem ocasionar desvios na determinação de grupos de objetos, bem como erros maiores na predição de novos valores, causados por valores inconsistentes (por conta do ruído) ou por falta de informação (com a ausência dos valores). Muitas técnicas de análise de dados não levam em conta tais questões, se valendo de processos separados que pré-processam a base de dados de forma a minimizar o efeito do ruído e a ausência dos dados, dentre os quais se destacam:

- **processamento de sinais:** para reduzir o ruído dos valores da base de tal forma que minimize uma possível interferência na análise;
- **seleção de atributos:** para descartar os atributos da base que não tenham significado prático

para a tarefa a ser realizada e que possam degradar o resultado da análise;

- **aprendizado de máquina:** para descobrir os possíveis motivos da ausência de dados, ou a origem do ruído, e realizar a imputação (estimação, determinação) de valores ou auxiliar a técnica de processamento de sinais.

Com isso, o pré-processamento dos dados, para que seja possível realizar a análise minimizando a influência das incertezas, requer a escolha de pelo menos três técnicas diferentes que realizem cada uma das tarefas listadas e, também, a escolha da ordem em que essas tarefas serão realizadas para obter o melhor desempenho final.

Alternativamente, pode ser vantajoso o uso de uma única técnica que realize as três tarefas simultaneamente, já levando em conta a possível interferência de uma tarefa nas demais. Uma técnica que possui essa propriedade é denominada *biclusterização* (Hartigan, 1972). Essa técnica consiste em, dentro de uma base de dados, encontrar subconjuntos de objetos e atributos da base de dados que reflitam relações significativas entre si. Dessa forma, a análise dos dados é feita apenas em sub-regiões da base original, realizando o procedimento de eliminar atributos redundantes, ou de pouca relevância, de acordo com o conjunto de objetos que está sendo avaliado no momento. Essa metodologia, capaz de gerar modelos locais dos dados, se contrapõe à clusterização, em que todos os objetos são avaliados levando em consideração todos os seus atributos. Embora essa seleção de atributos traga uma maior flexibilidade na análise dos dados, também traz uma maior complexidade computacional, uma vez que deve-se não apenas encontrar grupos de objetos inter-relacionados, como também descobrir em quais atributos essas relações se sustentam. Além disso, um mesmo objeto pode participar de múltiplos biclusters, com base em subconjuntos distintos de atributos.

Essa maior flexibilidade de análise promovida pela biclusterização também leva à possibilidade de entender melhor certos aspectos da base de dados que estão associados a sub-regiões de objetos e atributos, como, por exemplo, um ruído de medição que afeta somente alguns de muitos atributos, ou então apenas certos tipos de objetos dentre os analisados. Outro exemplo seria determinar o motivo da ausência dos valores de certos atributos, que podem estar associados aos valores de outros atributos e que ocorrem apenas em parte dos objetos.

Um dos tipos de bicluster dentre os que geralmente são estudados, o *bicluster coerente* (Cheng & Church, 2000), define cada bicluster como um modelo capaz de atenuar a influência do ruído associado à região da base de dados à qual ele pertence, através da minimização de um resíduo quadrático. Essas características da biclusterização se tornam interessantes no tratamento e análise de bases de dados com incertezas, seja na presença de ruídos ou na ausência de valores.

1.1 Objetivos e resultados esperados

Os objetivos desta tese podem ser resumidos nos seguintes pontos:

- estudar as propriedades das técnicas de biclusterização para realizar as três tarefas já descritas de pré-processamento de uma base de dados na presença de incerteza em seus valores;
- introduzir um novo algoritmo de biclusterização, focando nessas três tarefas;
- definir uma metodologia para utilizar as técnicas de biclusterização para a tarefa de imputação de dados faltantes.

Com o primeiro objetivo, é esperado aferir as propriedades desejadas que um bicluster deve possuir para ser capaz de realizar ou auxiliar na realização das tarefas de pré-processamento. Para isso, foi criada uma base de dados artificial contendo incertezas, controladas experimentalmente, simulando um evento real. Uma técnica de biclusterização foi aplicada junto a essa base de dados e os biclusters encontrados foram utilizados para avaliar tais propriedades.

O segundo objetivo torna-se necessário por causa da polarização dos estudos de biclusterização, que se concentram, em grande parte, na aplicação em análise de dados de expressão gênica (Agrawal et al., 1998; Cheng & Church, 2000; Tang et al., 2001), fazendo com que parte desses algoritmos tenham sido criados especificamente para essa única aplicação e deixando de evidenciar algumas propriedades desejáveis em um bicluster, que podem ser úteis para a análise de dados incertos.

Em relação ao terceiro objetivo, é necessário que, uma vez obtido um conjunto de biclusters, seja definido como esses biclusters podem ser utilizados para as diversas tarefas de análise e correção da incerteza nos dados. Idealmente, essa metodologia deve ser genérica o suficiente para ser utilizada por qualquer técnica de biclusterização, sendo limitada apenas pela qualidade dos biclusters encontrados.

1.2 Contribuições

Seguindo esses objetivos, esta tese traz diversas contribuições, tanto na área de biclusterização como na área de análise de dados com incertezas. Dentre elas, as principais são:

- criação de uma heurística construtiva capaz de encontrar um bicluster dentro de uma base de dados, respeitando a restrição de qualidade mínima enquanto maximiza o tamanho desse bicluster;
- utilização de uma meta-heurística bio-inspirada que é capaz de auxiliar e direcionar as escolhas da heurística acima citada, visando encontrar biclusters de melhor qualidade;

- proposição de uma meta-heurística com o objetivo de coordenar as heurísticas citadas nos itens anteriores, definindo um algoritmo de biclusterização capaz de encontrar um conjunto de biclusters que, além da otimização dos objetivos já citados, seja capaz de maximizar a região da base de dados que é representada por esse conjunto;
- determinação de uma metodologia para a utilização de técnicas de biclusterização na tarefa de imputação de dados, através da propriedade da biclusterização coerente em gerar modelos livres de ruído;
- formulação do problema de imputação de dados, em um bicluster coerente com valores faltantes, na forma de um problema de otimização quadrática, em que cada valor faltante se torna uma variável a ser otimizada;
- obtenção de fórmulas algébricas para calcular a derivada de cada uma das variáveis, levando a uma fórmula fechada necessária para viabilizar a solução do problema de otimização;
- prova matemática de que esse problema quadrático tem solução única e finita, contanto que uma condição seja satisfeita durante a obtenção do bicluster;
- resultados comparativos que mostram que o uso de biclusterização traz benefícios na qualidade da imputação, quando confrontado com outras técnicas, e na interpretabilidade dos dados.

1.3 Organização

A organização da tese foi feita de forma a primeiramente introduzir os conceitos do problema de dados faltantes, no Cap. 2. Em seguida, introduz-se e aprofunda-se o tema de biclusterização de dados, no Cap. 3. Define-se, no Cap. 4, um método de biclusterização capaz de maximizar a quantidade e diversidade de modelos extraídos da base de dados. Neste mesmo capítulo, algumas propriedades dos biclusters encontrados são avaliadas. Finalmente, no Cap. 5, é definido e demonstrado como realizar a imputação dos dados faltantes dentro dos biclusters encontrados, através da formulação de um problema de otimização quadrática. A solução deste problema conduz aos melhores valores para imputação, no sentido de que esses sejam os mais próximos possíveis dos valores livres de ruído. Neste mesmo capítulo, algumas propriedades obtidas por essa metodologia, e úteis para análise posterior dos dados, serão apresentadas. Considerações finais referentes aos resultados obtidos e às possíveis consequências futuras das contribuições desta tese serão descritas no Cap. 6, concluindo a tese e abrindo perspectivas para trabalhos futuros nessa linha de pesquisa.

Capítulo 2

O Problema dos dados faltantes

Na área de pesquisa de mineração de dados e extração de conhecimento (Han & Kamber, 2006), frequentemente supõe-se que os dados recebidos e que serão analisados estão corretos e completos. Mas isso costuma ser verdade apenas no caso de dados gerados artificialmente. Em bases de dados reais, é um fato comum a ocorrência de falhas no processo de aquisição da informação. Essas falhas podem ser de diferentes naturezas, como a aquisição incompleta de dados ou a presença de ruído no processo. Portanto, nessas situações, a qualidade da informação disponível é reduzida por essa imprecisão dos dados, podendo comprometer os resultados da análise.

Diversas aplicações demandam que esses dados sejam tratados para que se possa obter resultados coerentes, como em análise estatísticas ou análises de dados biológicos. Outras aplicações têm como objetivo estimar os dados faltantes de uma base, como é o caso de sistemas de recomendação (Adomavicius & Tuzhilin, 2005), em que é preciso, por exemplo, estimar o grau de satisfação de um cliente com determinado produto.

Em se tratando especificamente do problema da ausência de dados, referido a partir daqui como *problema de dados faltantes* (Longford, 2005), existem diversas causas possíveis para sua ocorrência. Exemplificando com a obtenção de dados estatísticos para uma pesquisa de opinião, o entrevistado pode se recusar a responder certas perguntas ou certas perguntas são preenchidas condicionalmente em relação às respostas anteriores; o entrevistador pode não realizar determinada pergunta por desatenção ou anotar de forma rasurada a resposta do entrevistado.

A ausência de respostas prejudica a análise dos dados estatísticos gerados, de modo que esses dados faltantes não podem ser ignorados. Essa ausência geralmente implica em um desequilíbrio nas propriedades estatísticas da amostra da população realizada, causando um viés na média das respostas. Nessa situação, é importante que a base de dados seja tratada de forma a manter essas propriedades estatísticas as mais próximas possíveis de seus valores reais.

Outro exemplo pode ser obtido em prontuários médicos contendo toda a relação de resultados

de exames de determinado paciente. Dependendo dos sintomas do paciente, este participará de um subconjunto de exames, mas dificilmente fará todos os exames possíveis. Armazenando os dados de todos os pacientes de um hospital em uma base de dados, levará fatalmente a muitos dados faltantes que, em alguma análise, poderão dificultar a obtenção de resultados ou ocasionar desvios nesses resultados.

Diversos problemas de classificação também sofrem frequentemente com dados faltantes. Considere, como exemplo, o problema padrão de classificar um determinado conjunto de objetos levando em conta um conjunto de atributos. Esses atributos podem ser comparados com uma base de dados de objetos pré-classificados de forma a encontrar o objeto que mais se assemelha àquele analisado, levando então à classificação do objeto desconhecido.

Porém, quando alguns desses atributos estão ausentes, o cálculo utilizado para verificar a semelhança entre objetos não levaria em conta tais atributos. Supondo que dois grupos distintos, dos possíveis grupos de classificação, possuam forte semelhança dentro do subconjunto de atributos não-ausentes no objeto, o que diferenciaria um objeto do outro são justamente os atributos que se encontram ausentes no objeto a ser classificado. Nessa situação, o método aplicado poderia classificar erroneamente objetos desses dois grupos. Para evitar isso, é necessário obter valores coerentes para tais objetos e que sejam os mais próximos possíveis dos valores reais.

Um problema mais prático na área de aprendizado de máquina é o denominado *Sistemas de Recomendação* (Adomavicius & Tuzhilin, 2005). Uma versão desse problema é: dada uma base de dados de histórico de consumo de produtos por uma lista de clientes e, em alguns casos, notas atribuídas aos produtos por esses clientes, o objetivo é gerar uma lista personalizada de recomendações de produtos para cada cliente, maximizando o acerto dessas recomendações.

Na sua forma mais básica, a base de dados desse sistema provém de duas fontes distintas: o ato de compra de um produto por um cliente e a avaliação de um produto por um cliente. É fácil observar que o principal motivo para a ausência de dados na primeira fonte de informação é a de que o cliente não consumiu ainda determinado produto ou consumiu de outro fornecedor. Já no que se refere à segunda fonte, somam-se a esses motivos os citados anteriormente em pesquisa de opinião.

O processo de recomendação é comumente tratado como um problema de classificação, ou seja, classificar e agrupar clientes similares, ou classificar e agrupar itens similares aos já consumidos por determinado cliente, para gerar as recomendações. Nesse caso, conforme mencionado anteriormente, a solução seria predizer os valores mais próximos aos valores que seriam atribuídos pelo próprio cliente. Porém, uma outra forma de desenvolver a recomendação é analisando e classificando o *motivo* de um determinado valor estar faltando na base de dados.

Para facilitar o entendimento, pode-se simplificar o problema levando em conta apenas dois motivos para um produto não ter sido consumido por determinado cliente: ele ainda não tem conhecimento

do produto ou ele conhece o produto e simplesmente não quer consumir. Identificar o motivo da ausência de cada um desses valores ajuda a filtrar apenas os produtos que devem participar do processo de classificação para recomendação. Isso pode levar a uma maior precisão, uma vez que o número de valores faltantes analisados pode ser reduzido. Uma forma de ser feito é procurar um padrão de comportamento, dentro da base de dados, que esteja relacionado com a ausência de determinados valores.

De acordo com as aplicações e a quantidade de valores faltantes na base de dados, diferentes métodos, com qualidades distintas, podem ser utilizados (McKnight et al., 2007):

(i) os valores faltantes podem ser ignorados quando correspondem a uma porcentagem muito pequena dos dados, ou quando a quantidade de dados é suficientemente grande para descartar os objetos contendo dados faltantes, sem perder as propriedades estatísticas.

(ii) os valores faltantes podem ser substituídos pelo valor médio do atributo correspondente, quando é importante ter alguma informação sobre o valor, mesmo que não seja muito precisa, e se esses valores faltantes correspondem a apenas uma pequena parcela da base de dados.

(iii) quando a informação faltante é realmente necessária ou a quantidade de dados ausentes representa uma alta porcentagem da base de dados, técnicas de aprendizado de máquina podem ser utilizadas para predizer e imputar esses valores.

Imputação de dados é o termo utilizado para a estimação e atribuição de um dado faltante, de forma que cause o menor impacto possível nas propriedades estatísticas reais da base de dados (Longford, 2005). Dada essa definição, quando os dados faltantes são causados por eventos conhecidos e que podem ser previstos, as técnicas de imputação empregadas são baseadas na *estimação de probabilidade* – em inglês, *likelihood estimation* (Little & Rubin, 2002).

O restante deste capítulo abordará com maior profundidade a teoria de dados faltantes, iniciando na terminologia geralmente utilizada (Longford, 2005; Donders et al., 2006; Graham, 2009) e, em seguida, focando nas causas mais comuns e sua classificação. Finalmente, as abordagens estatísticas de imputação serão apresentadas para cada uma das formas de dados faltantes, assim como técnicas alternativas provenientes da área de aprendizado de máquina.

2.1 Procedência dos dados

A necessidade de obtenção de dados vem com o propósito de ampliar o conhecimento disponível acerca de uma população de objetos. População é toda uma coleção bem definida de unidades, ou objetos, que possam ser determinados de forma inequívoca e sem ambiguidade. Por exemplo, pode-se

definir a população de todos os seres vivos do planeta Terra, ou ser mais específico como a população de todos os brasileiros homens com idade entre 18 e 25 anos. Note que, para determinar se um objeto pertence ou não a uma determinada população, sem ambiguidade, é necessário que ele contenha atributos que o caracterize. No segundo exemplo, os atributos poderiam ser *sexo*, *idade* e *nacionalidade*. Além dos atributos, geralmente os dados obtidos da população vêm com variáveis adicionais que são de interesse do estudo que se pretende realizar. Um exemplo para a população de brasileiros homens com idade entre 18 e 25 anos seria o valor do salário recebido atualmente.

Formas comuns de inferência em tarefas de análise estatística são cálculos como a média de determinado valor de subconjuntos da população, correlação entre os objetos da população ou a distribuição desses dados. No exemplo anterior, pode ser interessante ter conhecimento da média salarial da população, ou então identificar um perfil para os objetos da população que têm um salário acima da média, visando apontar fatores que contribua para a ocorrência de salários acima da média. Porém, para ser possível uma boa qualidade no processo de inferência, é necessário que a base de dados contenha uma grande quantidade de amostras da população dos objetos, o que frequentemente é inviável. Para citar alguns exemplos, um sensor de monitoramento adquire os dados com um intervalo de tempo entre uma leitura e outra; dados obtidos em sistemas de recomendação dependem da interação com o usuário; o processo de entrevista de uma população completa pode demandar mais tempo e dinheiro do que o disponível etc..

Dessa forma, o processo de inferência é feito apenas em uma amostra da população, isto é, um subconjunto de objetos da população original que retém propriedades de sua distribuição original. Seguindo o exemplo, a distribuição original da população pode ser 20% de pessoas com salário alto e 80% com salário médio ou baixo. Portanto, a amostra deverá manter essa mesma proporção. O processo de criação da amostra é determinado pelo *design de amostra* (π), que define a probabilidade de um objeto pertencer ou não à amostra.

Em uma população de tamanho N, objetos são retirados aleatoriamente dessa população e, de acordo com o design de amostra π, é sorteado se eles participarão da amostra ou não. Esse procedimento é efetuado até que se obtenham n objetos amostrados. Uma forma de avaliar se o design de amostra criado é adequado seria a utilização do erro das quantidades estatísticas geradas por ela. Uma *quantidade estatística* é alguma informação estatística aplicada à amostra como, por exemplo, a média de uma variável da amostra. Analogamente, *quantidade da população* é alguma informação estatística aplicada à população inteira.

Vamos tomar a média da população (μ) como exemplo, e a média amostral $\hat{\mu}$. O objetivo da criação do design de amostra é minimizar a seguinte equação:

$$|\hat{\mu} - \mu| \,. \tag{2.1}$$

Porém, como a geração da amostra é um processo estocástico, cada amostra pode obter um valor diferente para a Eq. 2.1. Portanto, uma forma de verificar a qualidade do design de amostra π é a partir de:

$$\lim_{H\to\infty} \frac{1}{H} \sum_{h=1}^{H} (\hat{\mu}_h - \mu)^2, \tag{2.2}$$

que nada mais é que a média do erro quadrático de infinitas amostras utilizando o design π. Obviamente é impossível executar esse processo infinitamente. Porém, pode-se determinar um valor alto de H para aumentar a precisão da medição do erro quadrático. Podemos definir também o *viés* do estimador como sendo a tendência da amostra estar fora da média, dado pela seguinte equação:

$$B(\hat{\mu}, \mu) = \frac{1}{H} \sum_{h=1}^{H} \hat{\mu}_h - \mu. \tag{2.3}$$

Essas duas equações são importantes para definir o objetivo de um design de amostra: minimizar o erro quadrático em relação à média e aproximar de zero o viés dessa amostra.

Juntamente com o problema de definir uma amostra que melhor represente a população ocorre o problema da ausência de dados ao realizar tal amostragem dos dados. Isso torna os objetivos descritos acima ainda mais difíceis de serem mensurados através das equações 2.2 e 2.3. Dentro das n amostragens da população, é possível obter tanto objetos completos como objetos incompletos (parcialmente ausentes) ou até mesmo objetos nulos (vazios). A incompletude dos dados pode ocorrer tanto durante o processo de amostragem ou ser inerente à população de onde a amostra foi retirada.

De qualquer forma, a teoria de amostragem apresentada serve para avaliar a melhor estimativa para esses valores de forma a maximizar a coerência nas quantidades estatísticas da base de dados. Isso define o problema dos dados faltantes: busca-se por um motivo para então fazer o processo de imputação de forma pertinente.

2.1.1 Presença de dados ruidosos

Um problema similar ao dos dados faltantes é a aquisição de dados ruidosos, ou seja, dados com valores alterados. Essa situação se assemelha à falta de dados pelo simples fato de que ela também modifica os parâmetros estatísticos da base de dados, podendo levar a problemas na análise ou classificação. Apesar da semelhança, esse problema é ainda mais complicado, pois não existe informação alguma de quais valores estão errados e da magnitude desse erro.

A existência de dados ruidosos pode ser atribuída a diversos fatores, como imprecisão de sensores ou aparelhos de medição ou falha de comunicação entre entrevistador e candidato em uma pesquisa de opinião. No caso de sistemas de recomendação, por exemplo, um cliente pode atribuir notas

diferentes a um mesmo produto se questionado em diferentes ocasiões.

2.2 Classificação dos dados faltantes

Conforme dito anteriormente, o problema de dados faltantes ocorre quando uma amostragem da população vem de forma incompleta. Apesar de essa ausência de dados geralmente ocorrer de forma aleatória, em alguns casos é possível, e pertinente, determinar os fatores que a ocasionaram, ou seja, a distribuição da probabilidade de ausência de um atributo. Essa determinação é importante para que a tarefa de tratamento dos dados, e possivelmente a imputação, se tornem mais precisas diante do conhecimento que deve ser extraído. A notação $(R|X^*)$ representa a distribuição aleatória das variáveis de tal forma que R se refere à distribuição dos valores obtidos e X^* ao conjunto de objetos da base. Define-se como X o conjunto de dados completos (com todos os valores em suas variáveis), e X_{falt} o conjunto de objetos faltantes, ou seja, a amostra de dados obtida contendo alguns objetos nulos ou incompletos. Seguindo essa notação, adotada na literatura de dados faltantes, é possível classificar um problema de dados faltantes segundo os motivos, ou mecanismos, da ausência dos dados (Longford, 2005; Donders et al., 2006; Graham, 2009).

O caso mais simples é denominado *MCAR*, do inglês *Missing Completely at Random*, e ocorre quando a causa da ausência dos dados é completamente aleatória, ou seja, independente de qualquer evento. Esse motivo pode ser representado por:

$$(R|X^*) \sim (R), \tag{2.4}$$

ou seja, a ocorrência de valores na base de dados não depende dos valores (ausentes ou não) dessa própria base. Por ser a causa mais simples, dada a ausência de fatores específicos que a produzem, frequentemente assume-se que o problema estudado é desse tipo para facilitar a obtenção de uma solução.

Um exemplo dessa situação ocorre quando uma pesquisa de opinião é feita aleatoriamente com um grupo de pessoas, sem levar em conta qualquer tendência, e uma porcentagem dessas não respondem à pesquisa ou respondem apenas parcialmente. Nesse caso específico, bastaria descartar esses dados incompletos ou refazer parte da pesquisa para completar os dados, supondo que a porcentagem de ausência aleatória é baixa.

Outro tipo de ausência de dados é denominado *MAR*, do inglês *Missing at Random*, que, apesar da nomenclatura levar a crer que é um caso similar ao MCAR, corresponde a dados que estão ausentes por causa de determinados valores em outros atributos. Ou seja, a probabilidade de um certo valor estar ausente é função dos valores já atribuídos às variáveis completas:

$$(R|X^*) \sim (R|X). \tag{2.5}$$

Esse caso é provavelmente o mais estudado em análises estatísticas, por capturar boa parcela da ausência de dados nessa área (Longford, 2005). Além disso, ele serve como simplificação de outro motivo de ausência de dados, o não-aleatório, que será explicado a seguir.

Um exemplo desse caso ocorre quando um médico está fazendo uma série de exames em um paciente, visando chegar a algum diagnóstico. Alguns dos exames devem ser feitos apenas se outros exames tiverem resultados positivos. É possível perceber que a ausência dos dados nesse caso (falta de resultados de certos exames) não é aleatória. Isso geralmente torna a situação mais complicada, uma vez que não é possível readquirir ou descartar os dados faltantes, pois eles fazem parte de um grupo com certa característica que deve ser levada em conta na análise dos dados.

Finalmente, o último tipo de dados faltantes é chamado de *NMAR*, do inglês *Not Missing at Random*, e se refere aos dados que estão ausentes por conta de seus próprios valores. Em outras palavras, a probabilidade de um valor estar ausente é igual à distribuição de seu próprio valor:

$$(R|X^*) \sim (R|X_{falt}). \tag{2.6}$$

Para exemplificar e melhorar o entendimento, em uma pesquisa de opinião, ao ser perguntada sobre sua renda familiar mensal, uma pessoa que tem um alto valor de rendimento pode se recusar a responder por questões de privacidade e segurança. O fato do valor desse atributo ser alto influenciou na decisão da pessoa não fornecer esse dado. Essa situação tende a ser mais complicada por não ser trivial determinar o motivo dessa ausência, uma vez que a explicação está contida no próprio valor ausente, tornando o processo de imputação mais trabalhoso. Algumas abordagens tentam transformar uma base com dados faltantes do tipo NMAR em uma base do tipo MAR, através de técnicas de aprendizado de máquina, para então realizar a imputação dos valores faltantes (Molenberghs et al., 2008).

A próxima seção introduzirá os métodos mais usuais de imputação de dados para essas situações, também levando em conta a aplicação que motiva essa imputação.

2.3 Métodos para lidar com dados faltantes

A escolha do método a ser empregado para lidar com a ausência de valores em uma base de dados é influenciada pela característica do problema e pela causa dos dados faltantes. A forma mais simples de resolver esse problema é eliminando todos os objetos da base que tenham informações incompletas ou nenhuma informação. Isso é possível nas situações em que a base contém muitos

objetos e a eliminação desses objetos não influencia a distribuição dos dados. No caso de problemas em que a população de objetos segue uma distribuição conhecida *a priori*, é possível remover os dados incompletos juntamente com outros dados completos, de forma a manter as propriedades da amostra. Isso é admissível desde que a quantidade de dados restantes não aumente o erro estatístico de forma significativa.

Quando a quantidade de dados não é suficiente para simplesmente eliminar os objetos incompletos, pode-se substituí-los por dados completos de mesma característica pertencentes à amostra, mantendo a proporção estatística estabelecida. A implicação dessa abordagem é um aumento no erro por utilizar dados redundantes. Mas, dependendo da aplicação, esse aumento pode ser aceitável, dada a simplicidade do método.

Note que os dois métodos mencionados têm um melhor funcionamento nas situações em que os dados faltantes são do tipo MCAR, pois, como a ausência dos dados ocorre de forma aleatória, as perdas de informação em cada atributo são reduzidas. Quando temos a situação de dados faltantes do tipo MAR, é necessário um cuidado maior, pois aqui há uma tendência de ocorrência da ausência em um atributo específico, reduzindo a informação existente para determinada faixa de valores desse atributo. Consequentemente, ao substituir os objetos incompletos ou eliminá-los, a base de dados adquire um caráter tendencioso para essa faixa de valores de atributos.

Para esses casos, é necessário que os valores faltantes sejam estimados e imputados na base de dados. A estimação dos dados tem que ser feita de tal forma que os valores estimados melhor se aproximem dos valores verdadeiros, ou então que as propriedades estatísticas da base de dados sejam as mais realistas possíveis.

2.3.1 Redução de dados

A técnica de redução de dados é um método que simplesmente descarta as amostras de objetos incompletos ou nulos. Essa técnica geralmente é empregada nos casos MCAR com poucos dados faltantes e também quando a amostra é grande o suficiente para não perder suas características ao ter as amostras incompletas retiradas. Apesar da simplicidade, essa técnica deve ser aplicada com cautela, verificando se a retirada dos objetos não afeta de modo significativo a distribuição das amostras. Outro cuidado que se deve ter é a decisão de remover objetos com pouca ausência de atributos, visto que os atributos completos desses objetos podem contribuir significativamente na inferência que se objetiva.

2.3.2 Métodos de imputação

Métodos de imputação, conforme dito anteriormente, tentam estimar os valores faltantes de forma que as propriedades estatísticas da base de dados se tornem as mais próximas possíveis das propriedades reais. Esses métodos podem ser classificados de diversas formas, como, por exemplo, levando-se em conta a área de atuação do método dentro da base de dados. Essa classificação possui três categorias: *imputação global nos atributos faltantes*, *imputação global nos atributos completos* e *imputação local* (Donders et al., 2006; Graham, 2009).

Os métodos de imputação global nos atributos faltantes geralmente usam uma medida de frequência para inferir os dados faltantes. Por exemplo, o valor mais frequente de um atributo específico pode ser utilizado para preencher os valores faltantes – técnica denominada *imputação pelo valor mais frequente de atributo*. Outro método é a imputação média, que simplesmente utiliza a média do valor dos atributos para preencher o espaço vazio, podendo também ser acrescido de uma perturbação aleatória para melhorar a variância dos resultados. Porém, da mesma forma que são métodos simples e computacionalmente eficientes, eles geralmente não trazem bons resultados quando comparados a outras metodologias, como apontado por Grzymala-Busse & Hu (2001); Scheffer (2002).

Os métodos de imputação global nos atributos completos são um conjunto de técnicas que exploram a correlação entre os atributos com valores faltantes e não-faltantes, e estimam os valores faltantes com regressões lineares ou logísticas. Essas técnicas devem considerar que a função de regressão utilizada reflete o modelo dos dados estudados e, por consequência, podem sofrer sobreajuste. Rubin (1987) propôs a técnica chamada *múltipla imputação*, que associa vários valores para cada valor faltante, estimados por modelos diferentes, avalia esses valores gerados e unifica tais resultados de um modo que gere uma melhor estimativa.

Esse método de múltipla imputação é separado em três passos distintos: *criação do conjunto de imputações*, *análise estatística das imputações* e *combinação dos resultados* (Longford, 2005).

Quanto aos métodos de imputação local, o mais famoso é denominado *Hot-Deck Imputation* (Ford, 1983; Sande, 1983), que preenche os valores faltantes com valores completos dessa base que contenham certa similaridade em outros atributos. Esse método primeiramente verifica a semelhança entre os objetos da base, utilizando apenas os atributos completos na comparação, e toma um conjunto de objetos similares para utilizar na imputação dos dados.

Outras formas de classificação de tratamentos de dados faltantes dizem respeito a como o método resolve o problema de dados faltantes: se é removendo os dados incompletos (*redução de dados*) ou completando os valores faltantes (*imputação de dados*). Finalmente cada método pode ser classificado também como *determinístico* ou *estocástico*, dependendo se os dados imputados serão sempre os mesmos, em uma mesma amostragem de dados, ou têm pequenas variações a cada vez que ele é aplicado.

Em seguida, maiores detalhes de algumas dessas técnicas serão apresentados. Análises mais detalhadas dessas técnicas podem ser encontradas em Longford (2005) e Lewis (1998).

Imputação pelo valor médio

A imputação pelo valor médio também é uma forma simples de imputar valores faltantes, atribuindo a média dos valores existentes de cada atributo em seus respectivos valores faltantes. Essa forma de imputação procura manter a média dos valores dos atributos de forma a causar pouco impacto nas análises estatísticas influenciadas pela média. Porém possui algumas desvantagens, como a perda da variabilidade *natural* dos valores de cada variável, tornando o desvio-padrão das variáveis abaixo do esperado. Isso, porém, pode ser corrigido somando o valor imputado a uma variável aleatória capaz de preservar o desvio-padrão dos valores do atributo. Essa imputação também não leva em conta possíveis relações entre o atributo incompleto e outros atributos completos. Exemplificando, imagine que uma amostra de dados tenha duas variáveis, uma relativa à altura de uma pessoa e outra relativa ao peso. Alguns valores referentes ao peso estão incompletos e a média que será imputada é de 70kg. Essa média pode ser viável para a maioria dos objetos que possuem a sua altura próxima da média, $1,75$m. Porém, em objetos cuja altura se distanciam da média (e.g.: $2,00$m), os valores imputados tendem a ficar abaixo dos valores esperados.

Imputação baseada em outro atributo

Outra forma simples de imputação, porém um pouco mais robusta, é analisar a relação entre dois atributos. Se for detectado que dois atributos têm valores similares, ou proporcionais, um desses atributos pode ser utilizado para imputar o valor do outro atributo. Essa forma de imputação tem a vantagem óbvia de utilizar uma informação mais precisa e coerente no momento de determinar o valor de um atributo. Porém, os casos em que é possível aplicar esse método não são comuns, uma vez que dependem da alta correlação entre variáveis e de uma inspeção na base de dados para encontrar essa correlação.

Vizinho mais próximo

O método de vizinho mais próximo procura pelo objeto completo mais similar ao objeto incompleto a ser imputado e copia seus valores como forma de imputação. Para implantar esse método, é necessário primeiramente definir uma função de *similaridade* que indique o quão similar são dois objetos, desconsiderando os valores faltantes. Esse método tem a clara vantagem de utilizar um objeto com características similares na hora da imputação e, portanto, tende a se aproximar mais do valor real do que os métodos mais simples descritos anteriormente.

Outra vantagem é na determinação da função de similaridade, que permite uma flexibilidade quanto à definição da importância relativa dos atributos do objeto. Por exemplo, em uma base cujos atributos são *sexo*, *idade*, *escolaridade*, *religião* e *salário*, o analista pode definir um peso maior para os atributos idade e escolaridade quando procurar por objetos similares visando imputar o atributo salário.

Apesar dessa possibilidade de flexibilização da função de similaridade, em certas ocasiões a definição dessa função pode ser uma tarefa desafiadora, particularmente quando os significados dos atributos e objetos não são muito bem definidos. Como exemplo, pode-se tomar uma base de dados de vídeos e os atributos de cada objeto seriam informações obtidas dos quadros pertencentes a esse vídeo. A definição da forma de comparação da similaridade entre dois vídeos é uma tarefa que requer amplos estudos de processamento de imagem.

Outra desvantagem dessa metodologia é referente ao desequilíbrio da qualidade de imputação. Isso ocorre porque, dependendo da amostra de dados, alguns objetos incompletos podem possuir um objeto muito similar que torne a imputação mais confiável, e outros objetos podem ser distantes de qualquer outro pertencente à amostra, de forma que a imputação tende a ser menos confiável.

Finalmente, o método de vizinho mais próximo não leva em conta, assim como a imputação pelo valor médio, a variabilidade do atributo. Porém, isso pode ser amenizado adotando a metodologia de *k-vizinhos mais próximos* (Beyer et al., 1999), onde os k objetos mais próximos ao objeto incompleto são utilizados para a imputação, permitindo incorporar alguma variabilidade.

Hot-deck

O método de *hot-deck* (Ford, 1983; Sande, 1983) é similar ao vizinho mais próximo, porém introduzindo estocasticidade. Esse método seleciona os n objetos mais similares ao objeto a ser imputado e, dentre esses, escolhe um aleatoriamente, de forma uniforme ou proporcional à similaridade entre os objetos, para efetuar a cópia dos valores faltantes em seu vetor de atributos.

Geralmente, esse processo é repetido sequencialmente para cada valor faltante em um objeto, sendo que probabilisticamente cada valor imputado tenderá a ser copiado de um objeto diferente, de forma a compensar a variabilidade no momento da cópia.

Uma vez que esse processo é baseado nos vizinhos mais próximos, ele também é afetado pelo problema de desequilíbrio na qualidade da imputação. Outro problema encontrado com esse método é a definição da quantidade de objetos selecionados: se o valor for muito alto, objetos pouco similares podem causar ruídos na imputação; se for muito pequeno, a variabilidade é afetada.

Regressão linear

Uma maneira de imputar os dados levando em conta a amostragem completa existente é através do método de regressão linear (Cohen, 2003). Esse método gera um modelo de predição capaz de encontrar uma relação entre os objetos completos (X) e os objetos incompletos (X_{falt}) como sendo:

$$X_{falt} = \beta_0 + \beta_1 X + \epsilon, \tag{2.7}$$

onde β_0 e β_1 são os pesos relativos à importância de cada objeto em relação a predição dos dados faltantes, e ϵ é um erro aleatório gerado para não reduzir a variância real dos valores dos atributos.

Um modelo linear de regressão mais generalizado gera uma combinação linear de diversas funções (lineares ou não), em torno dos objetos completos, de forma a aproximar o máximo possível a curva gerada dos pontos de amostragem da função-objetivo real X_{falt}. Esse modelo assume a seguinte forma:

$$X_{falt} = \sum_{j=0}^{m} \beta_j h_j(X) + \epsilon, \tag{2.8}$$

onde h_j é a j-ésima função-base com $h_0 = 1$, β_j é o peso atribuído à j-ésima função-base e m é o número de funções-base.

A Eq. 2.8 representa uma combinação linear de m funções-base, sendo que sua estrutura e parâmetros são pré-definidos. Mesmo que as funções-base sejam não-lineares, o modelo ainda é considerado linear nos parâmetros, pois a flexibilidade de obtenção do modelo está relacionada apenas à escolha dos coeficientes da combinação linear ($\beta_j, j = 1, ..., m$). Funções-base comumente usadas são polinomiais, wavelets, senóides e sigmóides. Os coeficientes da combinação linear são geralmente definidos com base em métodos de quadrados mínimos, e restrições de suavidade podem ser adicionadas ao modelo.

Esse método tem a clara vantagem de *aprender* um modelo que leva em conta a dependência dos atributos completos para inferir os valores incompletos, sendo então pertinente para valores faltantes do tipo MAR. Porém, se o modelo gerado se tornar muito específico às amostras, situação denominada *sobreajuste*, o resultado da imputação pode se tornar tendencioso em relação aos dados existentes.

Outra característica dessa abordagem é que ela pode ser feita com diversos conjuntos diferentes de amostras completas, selecionados através de algum critério pertinente à base de dados ou à variável a ser imputada, o que pode aliviar o efeito de um possível sobreajuste. Essa característica também é importante para o próximo método a ser apresentado: Múltipla Imputação.

Múltipla imputação

O objetivo principal do método de Múltipla Imputação (Ford, 1983; Sande, 1983; Rubin, 1987; Donders et al., 2006) é o de reduzir o viés no cálculo de quantidades estatísticas. Para isso ele gera diversos modelos de imputação para uma mesma base de dados utilizando algum método que não seja determinístico e, portanto, não apresente sempre o mesmo resultado para a mesma base de dados.

Um exemplo seria a utilização da regressão linear para a múltipla imputação. É possível escolher k conjuntos de amostras completas de forma que o vetor de pesos da combinação linear se torne distinto em cada um dos modelos. Consequentemente cada modelo irá gerar valores de imputação diferentes.

Ao final desse procedimento, existirão k bases completas com valores distintos de imputação. Nesse momento, ao realizar alguma medida estatística, as quantidades serão calculadas levando em consideração os valores de cada uma das bases. Exemplificando, o cálculo da média se torna:

$$\hat{\mu}_{MI} = \frac{1}{k}\sum_{i=1}^{k}\hat{\mu}_i, \tag{2.9}$$

onde $\hat{\mu}_{MI}$ é a estimativa da média estatística pela múltipla imputação e $\hat{\mu}_i$ é a média da base completa imputada i.

Similarmente, a variância estimada da base é definida como:

$$\hat{s}^2_{MI} = \bar{\hat{s}^2} + \frac{k+1}{k}\hat{B}, \tag{2.10}$$

onde

$$\bar{\hat{s}^2} = \frac{1}{k}\sum_{i=1}^{k}\hat{s}^2_i, \tag{2.11}$$

e

$$\hat{B} = \frac{1}{k-1}\sum_{i=1}^{k}(\hat{\mu}_i - \hat{\mu}_{MI})^2. \tag{2.12}$$

Esse método tem a clara vantagem de reduzir o viés da estimativa das quantidades estatísticas, além de reduzir o erro de estimativa e amenizar o problema de sobreajuste do método de regressão linear. Apesar disso, para obter uma estimativa correta é necessário um pré-estudo da base de dados visando identificar os modelos de imputação que melhor se adéque, além de buscar descobrir os tipos de dados faltantes e possíveis dependências entre os atributos.

Naïve Bayes

Outros métodos de imputação empregados são baseados em métodos tradicionais de aprendizado de máquina, como os métodos de aprendizado probabilístico. Esses métodos, no caso específico de predição de valores faltantes, são geralmente divididos em *métodos baseados em memória*, *métodos baseados em modelo* e *métodos livres de aprendizado* (Lemire, 2005). Os métodos livres de aprendizado são os mais simples de todos, geralmente empregando a média ponderada dos objetos ou atributos da base de dados para predizer os valores faltantes.

Os métodos baseados em modelo usam os parâmetros estatísticos dos dados existentes para aprender um modelo de predição. Métodos desse tipo geralmente utilizados compreendem Naïve Bayes, redes neurais e máquinas de vetores-suporte, dentre outros (Yu et al., 2004).

Finalmente, os métodos baseados em memória usam os dados de atributos de objetos completos similares para estimar os valores faltantes. Para determinar o conjunto de objetos similares que participarão na estimação de valores, é geralmente utilizado o método *k-vizinhos mais próximos* (k-VMP), já mencionado. Para a estimação dos dados, em sua forma mais comum, as médias ponderadas dos valores semelhantes são empregadas, levando em conta o desvio-padrão (Candillier et al., 2008). Porém, a partir dos conjuntos de objetos similares, outras técnicas de aprendizado podem ser efetuadas.

Voltando aos métodos baseados em modelo, além da vantagem de gerar um modelo único de predição, simplificando a tarefa de imputação de valores diante da possibilidade de surgimento de novos dados, algumas técnicas desse tipo podem também conter informações importantes sobre o modelo gerado, de tal forma que permita uma análise mais detalhada das propriedades da base de dados. Essa análise pode levar a aplicações de outros métodos para melhorar a imputação, como também gerar novos conhecimentos.

Um desses métodos baseados em modelo é denominado *Naïve Bayes*. Esse método tem como base o teorema de probabilidade condicional proposto por Bayes (1763) que, em termos simples, diz o seguinte:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}, \tag{2.13}$$

ou seja, a probabilidade da ocorrência do evento A dada a prévia ocorrência de um evento (ou conjunto de eventos) B é igual à probabilidade de ocorrer o evento B dada a pré-ocorrência de A juntamente com a probabilidade de ocorrer o evento A, normalizados pela probabilidade de ocorrer o evento B.

Esse teorema pode ser utilizado em imputação de dados empregando os objetos completos para construir um modelo de probabilidade condicional para aplicar aos dados faltantes. Uma forma de aplicar esse método pode ser exemplificada partindo da Tab. 2.1.

Nessa tabela, o valor referente ao atributo C do objeto 1 está faltando. Analisando essa base de

Tab. 2.1: Tabela de exemplo com 9 objetos e 3 atributos, onde o valor do primeiro objeto no atributo *C* está faltando.

	Atributos		
Objetos	**A**	**B**	**C**
1	2	1	–
2	3	1	2
3	2	2	1
4	2	3	3
5	2	1	2
6	2	1	1
7	1	2	3
8	2	1	2
9	1	3	1

dados, é possível perceber que os valores das variáveis estão dentro da faixa de $[1; 3]$. Então, para estimar o valor faltante, verifica-se a probabilidade de que o valor real dessas variáveis seja um dos possíveis valores, ou seja, calcula-se:

$$P_1(C = i \mid A = 2, B = 1), \tag{2.14}$$

com i variando de 1 a 3. A equação é lida como a probabilidade do objeto 1 possuir o valor i no atributo C dados os valores desse objeto nos atributos A e B. Essa probabilidade pode ser calculada da seguinte forma:

$$P_1(C = 1 \mid A = 2, B = 1) = \frac{P_1(A = 2, B = 1 \mid C = 1)P_1(C = 1)}{P_1(A = 2, B = 1)}. \tag{2.15}$$

Os valores das probabilidades do lado direito da equação podem ser calculados diretamente da base de dados, utilizando os objetos completos. Por exemplo:

$$\begin{aligned} P_1(A = 2, B = 1 \mid C = 1) &= \frac{1}{3} \\ &\cong 0,33, \end{aligned} \tag{2.16}$$

$$\begin{aligned} P_1(C = 1) &= \frac{3}{8} \\ &= 0,375, \end{aligned} \tag{2.17}$$

e

$$P_1(A = 2, B = 1) = \frac{3}{8} = 0,375, \tag{2.18}$$

substituindo-os na Eq. 2.15 temos:

$$P_1(C = 1 \mid A = 2, B = 1) \cong \frac{0,33 \cdot 0,375}{0,375} \cong 0,33. \tag{2.19}$$

De forma similar, verificamos a probabilidade do atributo C do objeto 1 ter os valores 2 e 3, com os seguintes resultados:

$$P_1(C = 2 \mid A = 2, B = 1) = \frac{0,66 \cdot 0,375}{0,375} = 0,66 \tag{2.20}$$

$$P_1(C = 3 \mid A = 2, B = 1) = \frac{0 \cdot 0,25}{0,375} = 0. \tag{2.21}$$

Utilizando essas informações, poderíamos imputar o valor de maior probabilidade ou da média ponderada. Esse método é vantajoso quando a amostra de objetos completos é grande o suficiente para a construção de um modelo de probabilidade coerente. O modelo de probabilidade não só serve para imputar os dados, mas também para tentar descobrir o que ocasionou a falta dos mesmos, ajudando na detecção do tipo de dados faltantes referente à base de dados. Como desvantagens, esse método, do jeito mais simples aqui apresentado, pode não ser robusto em relação ao ruído contido na base, causando incoerências no momento das inferências. Outro problema é quando os atributos possuem valores contínuos, tornando necessária uma divisão em faixas de valores, o que causa imprecisão no resultado final da imputação.

Apesar disso, diversas versões desse método foram criadas para tentar reduzir o efeito dessas desvantagens e tornar o método Naïve Bayes aplicável a situações reais (Lewis, 1998).

2.4 Exemplo didático

Nesta seção, uma base de dados artificial é criada, denominada *ArtDataset*, sobre a qual os métodos apresentados serão aplicados e avaliados através de diversos objetivos, de forma que seus potenciais e limitações sejam evidenciados. Essa base de dados inicialmente é gerada supondo exatidão na aquisição dos dados, ou seja, sem a presença de ruído.

Para tornar possível a aplicação das metodologias descritas anteriormente, essa base de dados passou por um processamento de forma a torná-la parecida com uma situação real e comum em se tratando de dados ruidosos. Essa base contém 1000 objetos e 100 atributos ou variáveis. Esses atributos podem conter valores na faixa de $[1, 5]$ e podem ser interpretados como a opinião, através de notas, de entrevistados junto a quatro assuntos distintos, cada qual com 25 questões, totalizando as 100 variáveis.

Um objeto é dito pertencer a um determinado grupo de interesse em um dos assuntos se ele atribuiu nota igual ou superior a quatro na maior parte dos atributos desse assunto, e isso pode ser verificado por métodos de agrupamento abordados posteriormente no Cap. 3. Cada objeto pode pertencer a mais de um grupo ao mesmo tempo.

Note que essa base de dados serve apenas como forma de ilustrar as características de cada método, além de verificar propriedades que possam ser úteis em análises mais aprofundadas ou formas distintas de extrair conhecimento da base. Bases de dados reais também serão utilizadas neste trabalho, quando for pertinente, para atestar a real capacidade de alguns métodos apresentados em situações típicas.

A partir dessa base serão construídas outras quatro bases de dados artificiais: uma base de dados ruidosa, denominada *ArtDatasetRuido*; uma base de dados com dados faltantes do tipo MCAR, denominada *ArtDatasetMCAR*; uma base de dados com dados faltantes do tipo MAR, denominada *ArtDatasetMAR*; e uma base de dados com dados faltantes do tipo NMAR, denominada *ArtDatasetNMAR*. Os detalhes da construção dessa base de dados estão presentes no Apêndice A.

2.5 Comparativo entre algumas metodologias

De forma a comparar o desempenho e as diferentes propriedades das metodologias apresentadas neste capítulo, é necessário primeiro definir alguns tratamentos estatísticos. Esses tratamentos serão divididos em tarefas, as quais deverão refletir possíveis objetivos de extração de conhecimento da base de dados, levando em conta o desafio imposto pela ausência de dados.

As tarefas serão separadas de acordo com a base de dados que será utilizada, pois cada uma remete a uma série de desafios distintos. Como base de comparação, serão utilizadas a base original, ArtDataset, e a base ruidosa, ArtDatasetRuido. Apesar da tendência dos algoritmos em se aproximarem da base ruidosa, é interessante notar se alguma metodologia consegue abstrair parte do ruído e tender às quantidades estatísticas da base original.

A primeira tarefa, mais simples de todas, é calcular média e desvio-padrão das bases completas, com e sem ruídos. Essa tarefa traz uma visão geral do erro de estimação. Como segunda tarefa, é feito o cálculo da média e desvio-padrão para cada grupo de usuários. Essa tarefa é aplicada para a

base de dados ArtDatasetMCAR e ArtDatasetMAR, mas não para a base ArtDatasetNMAR, pois ela é pouco afetada nessa medição. A terceira tarefa é específica para a base de dados ArtDatasetMAR e corresponde ao cálculo da média e desvio-padrão dos atributos 65, 80 e 95, que são os atributos incompletos. Similarmente, para a base de dados ArtDatasetNMAR, é dada a tarefa de calcular a média e desvio-padrão dos atributos 30 e 85.

Além dessas análises, também foram calculados os erros de estimação através do erro quadrático médio (EQM) e do erro absoluto médio (EAM), descritos nas Eqs. 2.22 e 2.23, respectivamente, e levando em conta as bases de dados ArtDataset e ArtDatasetRuido. O cálculo do EQM é dado por:

$$EQM = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2, \tag{2.22}$$

onde N é o número de valores estimados, x_i é o i-ésimo valor real e $\hat{x}_i$ é o i-ésimo valor estimado. Analogamente, o cálculo do EAM é dado por:

$$EAM = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x}_i|. \tag{2.23}$$

A medição desses erros servirá para determinar o quanto cada método se aproximou da base de dados apresentada. A Tab. 2.2 mostra os valores de cada tarefa para as bases ArtDataset e ArtDatasetRuido, que serão utilizados para comparação com os resultados obtidos em cada metodologia.

Tab. 2.2: Valores das quantidades estatísticas pertinentes a cada tarefa. Os símbolos μ e σ representam a média e o desvio-padrão, respectivamente. Os índices $g1$, $g2$, $g3$ e $g4$ se referem a quantidades estatísticas calculadas nos subconjuntos definidos pelos grupos 1, 2, 3 e 4, respectivamente. Finalmente, os índices numéricos remetem a quantidades estatísticas em um determinado atributo em todo o conjunto de objetos.

		Bases de Dados	
	Tarefa	**ArtDataset**	**ArtDatasetRuido**
1	$\mu \pm \sigma$	$2,70 \pm 1,49$	$2,70 \pm 1,79$
2	$\mu_{g1} \pm \sigma_{g1}$	$4,52 \pm 0,21$	$4,51 \pm 1,02$
3	$\mu_{g2} \pm \sigma_{g2}$	$4,51 \pm 0,22$	$4,51 \pm 1,03$
4	$\mu_{g3} \pm \sigma_{g3}$	$4,48 \pm 0,23$	$4,48 \pm 1,02$
5	$\mu_{g4} \pm \sigma_{g4}$	$4,51 \pm 0,20$	$4,51 \pm 1,01$
6	$\mu_{65} \pm \sigma_{65}$	$2,64 \pm 1,47$	$2,63 \pm 1,78$
7	$\mu_{80} \pm \sigma_{80}$	$3,02 \pm 1,43$	$3,00 \pm 1,72$
8	$\mu_{95} \pm \sigma_{95}$	$3,01 \pm 1,41$	$3,00 \pm 1,71$
9	$\mu_{30} \pm \sigma_{30}$	$2,79 \pm 1,52$	$2,86 \pm 1,82$
10	$\mu_{85} \pm \sigma_{85}$	$2,97 \pm 1,37$	$2,98 \pm 1,71$

2.5.1 Resultados

Os experimentos comparativos foram conduzidos para os algoritmos de Redução de dados (RD), Imputação pelo valor médio (IVM), Imputação baseada em outro atributo (IAT), Vizinho mais próximo (VMP), Hot-deck (HD), Regressão linear (RL) e Múltipla imputação (MI). Para o algoritmo VMP, foi utilizada a distância euclidiana como medida de similaridade. No caso do HD, foi adotada a mesma medida e um número de 200 objetos para serem escolhidos, definido empiricamente. O RL utilizou um procedimento simples de regressão linear múltipla, baseada em quadrados mínimos (Chatterjee & Hadi, 1986). Finalmente, a MI utiliza essa mesma regressão linear, porém empregando quatro amostras diferentes, uma para cada grupo em que a base foi dividida. Note que, em uma situação real, nem sempre é possível ter o conhecimento de quantos e quais grupos existem na base de dados.

Na Tab. 2.3, é possível observar os resultados das quantidades estatísticas para as tarefas propostas na base de dados ArtDatasetMCAR, onde os dados faltantes são do tipo MCAR. Note que os valores apresentados são arredondados para duas casas decimais, o que implica que dois valores iguais podem ser, na verdade, muito próximos entre si. A primeira observação a ser feita é que, uma vez que todos os objetos estavam incompletos, a técnica RD foi incapaz de produzir qualquer resultado. Na primeira tarefa, é possível perceber que o único a obter aproximadamente a mesma média da base de dados original foi a técnica IVM, porém, o desvio-padrão ficou abaixo do real. As outras abordagens apresentaram uma tendência oposta ao IVM, aproximando-se mais do desvio-padrão e se afastando da média. Vale notar que a abordagem IAT teve o melhor desempenho nessa tarefa, obtendo aproximadamente o mesmo desvio-padrão, porém, com a média um pouco abaixo.

Para a segunda tarefa, ocorre algo diferente; duas técnicas, IAT e VMP, se aproximam tanto da média real quanto do desvio-padrão. Para o caso da IAT, ela se aproximou um pouco mais do desvio-padrão, enquanto a VMP ficou mais próxima à média real. As outras abordagens tiveram uma dificuldade maior quanto ao desvio. Na terceira tarefa, a técnica IAT dessa vez foi a que melhor se aproximou da média, enquanto a MI chegou mais próxima ao desvio-padrão real, sem comprometer o valor da média. Na quarta tarefa, a técnica IAT melhor se aproximou tanto da média quanto do desvio-padrão, assim como na quinta tarefa. Porém, na quinta tarefa, a técnica RL obteve a mesma distância em relação aos valores reais do desvio-padrão.

Na sexta tarefa, novamente IAT obteve os melhores resultados. Nos resultados da sétima tarefa, a melhor média é obtida pela técnica IVM, enquanto o melhor desvio foi obtido pela MI. As técnicas IAT e RL obtiveram um melhor equilíbrio entre os dois resultados. Na oitava tarefa, a técnica IVM novamente apresenta a melhor média, enquanto a técnica RL apresenta o melhor desvio-padrão, sem comprometer o valor da média.

Na nona tarefa, verifica-se que a técnica MI obtém média e desvio-padrão mais próximos, sendo

seguida pelas técnicas IAT e HD, que obtiveram uma média um pouco maior. Finalmente, na décima tarefa, embora a técnica MI tenha melhor desvio-padrão, as técnicas IAT e RL obtiveram maior êxito com a média, sem comprometer o desvio.

Os resultados para a base de dados com valores faltantes do tipo MAR, denominada aqui ArtDatasetMAR, são apresentados na Tab. 2.4. Nessa tabela, é possível perceber que os resultados da primeira tarefa são pouco afetados por conta da baixa quantidade de valores faltantes e uma melhor estimação. Também é possível notar que a técnica RD, por remover objetos incompletos, retira parte importante dos dados, causando um erro significativo na média.

Uma vez que os dados faltantes do tipo MAR estão presentes nos grupos 3 e 4, as tarefas correspondentes a esses grupos apresentam resultados idênticos, com exceção novamente da técnica RD. Nas duas tarefas seguintes, com exceção das técnicas RD e IVM, todas apresentam um desempenho satisfatório, valendo mencionar que as técnicas IAT e RL obtiveram a melhor aproximação e a técnica MI teve apenas uma diferença baixa no desvio-padrão da tarefa correspondente ao grupo 3.

Na tarefa correspondente ao atributo 65, as técnicas IAT e RL foram as que mais se aproximaram do resultado real. Para a tarefa do atributo 80, todas as abordagens conseguiram um bom desempenho, inclusive a RD. Porém, a que obteve maior aproximação foi a técnica IAT. Finalmente, na tarefa do atributo 95, as técnicas IVM e IAT se aproximaram mais do valor real, embora as outras técnicas não tenham obtido valores muito distantes. As duas tarefas finais foram omitidas pois nelas faltam dados apenas dos tipos MCAR e NMAR e, portanto, não afetam em nada esse experimento.

Finalmente, na Tab. 2.5, são apresentados os resultados para os experimentos na base ArtDatasetNMAR, correspondentes a valores faltantes do tipo NMAR. Novamente, os valores da média e desvio-padrão da base completa estão muito próximos do valor real. Isso ocorre justamente pelo menor número de valores faltantes. Com os valores de precisão utilizados nesse experimento, aparentemente nenhum dos grupos teve seus valores estatísticos afetados, exceto para RD, onde ocorre redução de dados.

Quanto aos atributos específicos ao NMAR, o atributo 30 foi melhor estimado pelo RL com relação à média e desvio-padrão. Para o atributo 85, o algoritmo RL foi aquele que melhor aproximou a estimativa nas quantidades estatísticas.

A próxima análise foi feita em relação ao quanto os dados estimados se aproximaram das bases reais. Note que, apesar de intuitivamente se supor que os métodos que melhor estimaram os valores estatísticos terão os menores erros, isso não é sempre verdade. Um método pode possuir um erro menor do que o de outro, porém com um viés maior, fazendo com que modifique as propriedades estatísticas da base de dados.

Na Tab. 2.6, estão apresentados os valores dos erros para a base ArtDatasetMCAR das estimativas feitas por cada método. Com os resultados, é possível perceber que os métodos RL e MI apresentaram

Tab. 2.3: Resultados de estimação dos valores das quantidades estatísticas pertinentes a cada tarefa produzidas pelo conjunto de algoritmos estudados neste capítulo, aplicados à base de dados ArtDatasetMCAR, com valores faltantes do tipo MCAR. Os valores da tabela foram obtidos após a execução de cada algoritmo de tratamento de dados faltantes.

		Bases de Dados ArtDatasetMCAR						
	Tarefa	**RD**	**IVM**	**IAT**	**VMP**	**HD**	**RL**	**MI**
1	$\mu \pm \sigma$	–	$2,70 \pm 1,71$	$2,69 \pm 1,79$	$2,65 \pm 1,79$	$2,66 \pm 1,78$	$2,69 \pm 1,77$	$2,67 \pm 1,78$
2	$\mu_{g1} \pm \sigma_{g1}$	–	$4,30 \pm 1,17$	$4,49 \pm 1,03$	$4,51 \pm 1,00$	$4,37 \pm 1,20$	$4,49 \pm 0,99$	$4,41 \pm 1,15$
3	$\mu_{g2} \pm \sigma_{g2}$	–	$4,34 \pm 1,10$	$4,49 \pm 1,05$	$4,20 \pm 1,39$	$4,35 \pm 1,22$	$4,48 \pm 0,99$	$4,48 \pm 1,04$
4	$\mu_{g3} \pm \sigma_{g3}$	–	$4,31 \pm 1,10$	$4,48 \pm 1,02$	$4,23 \pm 1,28$	$4,20 \pm 1,34$	$4,47 \pm 0,98$	$4,42 \pm 1,09$
5	$\mu_{g4} \pm \sigma_{g4}$	–	$4,37 \pm 1,06$	$4,49 \pm 1,03$	$4,19 \pm 1,44$	$4,23 \pm 1,33$	$4,49 \pm 0,98$	$4,44 \pm 1,09$
6	$\mu_{65} \pm \sigma_{65}$	–	$2,60 \pm 1,70$	$2,63 \pm 1,79$	$2,49 \pm 1,74$	$2,50 \pm 1,76$	$2,62 \pm 1,76$	$2,59 \pm 1,79$
7	$\mu_{80} \pm \sigma_{80}$	–	$3,01 \pm 1,65$	$2,97 \pm 1,73$	$2,88 \pm 1,71$	$2,87 \pm 1,75$	$2,97 \pm 1,71$	$2,95 \pm 1,72$
8	$\mu_{95} \pm \sigma_{95}$	–	$3,00 \pm 1,64$	$2,99 \pm 1,73$	$2,57 \pm 2,12$	$2,85 \pm 1,73$	$2,99 \pm 1,70$	$2,96 \pm 1,72$
9	$\mu_{30} \pm \sigma_{30}$	–	$2,88 \pm 1,73$	$2,87 \pm 1,83$	$2,58 \pm 1,93$	$2,87 \pm 1,83$	$2,83 \pm 1,80$	$2,85 \pm 1,82$
10	$\mu_{85} \pm \sigma_{85}$	–	$2,96 \pm 1,63$	$2,96 \pm 1,72$	$2,79 \pm 1,72$	$2,85 \pm 1,70$	$2,96 \pm 1,69$	$2,92 \pm 1,71$

Tab. 2.4: Resultados de estimação dos valores das quantidades estatísticas pertinentes a cada tarefa produzidos pelo conjunto de algoritmos estudados neste capítulo, aplicados à base de dados ArtDatasetMAR, com valores faltantes do tipo MAR.

		Bases de Dados MAR						
	Tarefa	**RD**	**IVM**	**IAT**	**VMP**	**HD**	**RL**	**MI**
1	$\mu \pm \sigma$	$2,64 \pm 1,78$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$
2	$\mu_{g1} \pm \sigma_{g1}$	$4,49 \pm 0,98$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$
3	$\mu_{g2} \pm \sigma_{g2}$	$2,80 \pm 1,97$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$
4	$\mu_{g3} \pm \sigma_{g3}$	$2,91 \pm 1,92$	$4,31 \pm 1,02$	$4,48 \pm 1,02$	$4,47 \pm 1,04$	$4,47 \pm 1,04$	$4,48 \pm 1,02$	$4,48 \pm 1,03$
5	$\mu_{g4} \pm \sigma_{g4}$	$2,98 \pm 1,81$	$4,37 \pm 1,01$	$4,51 \pm 1,01$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,01$	$4,51 \pm 1,01$
6	$\mu_{65} \pm \sigma_{65}$	$2,56 \pm 1,78$	$2,60 \pm 1,73$	$2,63 \pm 1,79$	$2,48 \pm 1,75$	$2,48 \pm 1,76$	$2,63 \pm 1,77$	$2,59 \pm 1,80$
7	$\mu_{80} \pm \sigma_{80}$	$3,01 \pm 1,72$	$3,01 \pm 1,70$	$3,00 \pm 1,72$	$3,01 \pm 1,71$	$3,01 \pm 1,71$	$3,01 \pm 1,71$	$3,01 \pm 1,71$
8	$\mu_{95} \pm \sigma_{95}$	$3,01 \pm 1,71$	$3,00 \pm 1,71$	$3,00 \pm 1,71$	$2,97 \pm 1,75$	$2,99 \pm 1,72$	$3,01 \pm 1,71$	$3,01 \pm 1,71$

Tab. 2.5: Resultados de estimação dos valores das quantidades estatísticas pertinentes a cada tarefa produzidos pelo conjunto de algoritmos estudados neste capítulo, aplicados à base de dados ArtDatasetNMAR, com valores faltantes do tipo NMAR.

		Bases de Dados NMAR						
	Tarefa	**RD**	**IVM**	**IAT**	**VMP**	**HD**	**RL**	**MI**
1	$\mu \pm \sigma$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$	$2,70 \pm 1,79$
2	$\mu_{g1} \pm \sigma_{g1}$	$3,79 \pm 1,95$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$	$4,51 \pm 1,02$
3	$\mu_{g2} \pm \sigma_{g2}$	$4,70 \pm 0,99$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$	$4,51 \pm 1,03$
4	$\mu_{g3} \pm \sigma_{g3}$	$3,09 \pm 1,70$	$4,48 \pm 1,02$	$4,48 \pm 1,02$	$4,48 \pm 1,02$	$4,48 \pm 1,02$	$4,48 \pm 1,02$	$4,48 \pm 1,02$
5	$\mu_{g4} \pm \sigma_{g4}$	$3,17 \pm 1,62$	$4,51 \pm 1,01$	$4,51 \pm 1,01$	$4,50 \pm 1,03$	$4,50 \pm 1,03$	$4,51 \pm 1,01$	$4,51 \pm 1,01$
9	$\mu_{30} \pm \sigma_{30}$	$2,89 \pm 1,83$	$2,89 \pm 1,81$	$2,86 \pm 1,83$	$2,83 \pm 1,85$	$2,88 \pm 1,83$	$2,86 \pm 1,82$	$2,86 \pm 1,83$
10	$\mu_{85} \pm \sigma_{85}$	$2,90 \pm 1,71$	$2,91 \pm 1,68$	$2,96 \pm 1,71$	$2,84 \pm 1,72$	$2,87 \pm 1,71$	$2,97 \pm 1,70$	$2,94 \pm 1,70$

o menor erro, em todas as situações apresentadas, embora, na média, o método IAT tenha obtido melhores aproximações dos valores estatísticos, conforme visto anteriormente.

Tab. 2.6: Erro absoluto médio (EAM) e erro quadrático médio (EQM) dos valores estimados por cada algoritmo para a base de dados ArtDatasetMCAR, em relação à base original sem ruído e à base ruidosa, com a qual foram feitos os experimentos de imputação.

ArtDatasetMCAR	**ArtDataset**		**ArtDatasetRuido**	
Algoritmo	**EAM**	**EQM**	**EAM**	**EQM**
IVM	1,32	2,04	1,42	2,99
IAT	0,78	1,06	1,09	2,02
VMP	1,90	5,77	2,07	6,72
HD	1,74	4,99	1,91	5,92
RL	0,57	0,53	0,96	1,22
MI	0,60	0,61	0,97	1,57

A Tab. 2.7 apresenta os valores dos erros para a base ArtDatasetMAR das estimações feitas por cada método. Aqui é possível perceber que, como a ausência dos dados estava distribuída entre apenas três atributos, o erro médio gerado tende a ser maior, dado que a estimação contém menos informação. Novamente, os métodos RL e MI obtiveram os menores valores de erro, seguidos do método IAT. Nesse experimento, é possível notar também que os erros em relação à base original, sem ruído, é menor, indicando que os valores imputados tendem aos valores reais, livres de ruídos.

Tab. 2.7: Erro absoluto médio (EAM) e erro quadrático médio (EQM) dos valores estimados por cada algoritmo para a base de dados ArtDatasetMAR, em relação à base original sem ruído e à base ruidosa, com a qual foram feitos os experimentos de imputação.

ArtDatasetMAR	**ArtDataset**		**ArtDatasetRuido**	
Algoritmo	**EAM**	**EQM**	**EAM**	**EQM**
IVM	1,75	3,16	1,80	4,04
IAT	0,89	1,06	1,07	1,91
VMP	2,59	9,38	2,65	9,61
HD	2,64	8,77	2,70	9,20
RL	0,85	0,94	1,02	1,62
MI	0,87	0,96	1,05	1,71

Finalmente, na Tab. 2.8, os valores dos erros para a base ArtDatasetNMAR das estimações feitas por cada método são comparados. Uma vez que a ausência dos dados não tem relação com nenhum outro atributo, o processo de estimação torna-se mais desafiador, elevando os erros médios. Porém, o número menor de valores faltantes resultou em um erro menor, para alguns dos métodos, quando

comparado com os problemas anteriores. É possível notar, em especial, os erros maiores das técnicas IVM, VMP e HD, que pressupõem uma semelhança entre os valores de todos os atributos. Novamente, as técnicas que obtiveram melhor desempenho foram a MI e a RL.

Tab. 2.8: Erro absoluto médio (EAM) e erro quadrático médio (EQM) dos valores estimados por cada algoritmo para a base de dados ArtDatasetNMAR, em relação à base original sem ruído e à base ruidosa, com a qual foram feitos os experimentos de imputação.

ArtDatasetNMAR	**ArtDataset**		**ArtDatasetRuido**	
Algoritmo	**EAM**	**EQM**	**EAM**	**EQM**
IVM	1,01	1,30	1,62	2,04
IAT	0,99	1,28	1,53	1,89
VMP	1,14	1,47	1,71	2,20
HD	1,12	1,45	1,73	2,21
RL	0,95	1,22	1,55	1,98
MI	0,97	1,26	1,57	2,01

2.6 Sistemas de Recomendação

Um caso particular e interessante no estudo de ausência de dados é a classe de algoritmos denominada Sistemas de Recomendação (Adomavicius & Tuzhilin, 2005). Esses sistemas são definidos basicamente como mecanismos capazes de extrair informações comportamentais de usuários de forma a recomendar itens que tenham alta probabilidade de serem aceitos por tais usuários.

As informações comportamentais podem se basear em histórico de consumo de objetos, índice de satisfação sobre os produtos consumidos, características regionais e sócio-econômicas de cada usuário, dentre outros. Os itens podem ser produtos de consumo de diversas espécies.

Os sistemas de recomendação se dividem em três tipos diferentes (Adomavicius & Tuzhilin, 2005): (i) *baseado em conteúdo*, onde toda informação passada ou atual sobre o produto e o cliente é utilizada individualmente para inferir recomendações; (ii) *filtragem colaborativa*, que usa a informação contida na relação item-usuário, geralmente dada em forma de avaliação dos usuários aos produtos consumidos, para gerar modelos de itens e usuários similares entre si, em relação às suas propriedades ou preferências; (iii) *híbrida*, que combina os dois tipos anteriores de forma a suprir as deficiências de cada um deles.

Para auxiliar na compreensão das técnicas de sistemas de recomendação, um exemplo ilustrativo será dado. Nesse exemplo, o sistema de recomendação será responsável por recomendar filmes a clientes de uma locadora. Os objetos desse sistema são representados pelos filmes e os usuários pelos

clientes da locadora. Quanto aos filmes, o sistema tem informações sobre sinopse, gênero, diretor, atores. Dos usuários, o sistema tem conhecimento sobre a região onde o cliente mora e histórico de locações.

Com essas informações, o sistema de recomendação baseado em conteúdo pode empregar técnicas de mineração de dados e recuperação de informação para encontrar filmes que tendem a produzir o mesmo grau de satisfação obtido com aqueles que o cliente assistiu previamente e gostou. Naturalmente, os atendentes da locadora já fazem esse tipo de recomendação, lembrando filmes que o cliente levou recentemente e gostou, e recomendando aqueles que eles julgam produzir o mesmo grau de satisfação. Esse tipo de recomendação ajuda a encontrar o gosto *generalizado* do cliente. Porém, o cliente pode estar com intenção de alugar filmes de outro gênero diferente do que está acostumado, buscando sensações alternativas.

Nessa situação, o sistema de recomendação poderia verificar qual filme o cliente está observando nesse exato momento e, então, encontrar outros filmes com conteúdo similar para recomendá-los. Dessa forma, a recomendação refletiria a intenção atual do cliente, em vez de seu comportamento médio.

Esse sistema pode também realizar uma recomendação indireta, primeiramente procurando outros clientes com atributos similares e, então, utilizar o histórico de locação desses para gerar a lista de recomendação.

A utilização desse método implica que é possível comparar dois itens através de uma métrica simples, que retorne o grau de similaridade, e que essa similaridade seja significativa. Um ponto a ser levado em conta ao implementar tal técnica é que, embora o conteúdo descritivo dos itens seja similar, não implica que a qualidade dos dois seja igualmente similar. Por exemplo, produtos iguais de marcas diferentes. Outro ponto é que esse sistema não é capaz de detectar similaridades implícitas entre diferentes categorias de objetos. Por exemplo, associar a compra de um aparelho eletrônico às pilhas requeridas por ele.

Os sistemas de filtragem colaborativa (Herlocker et al., 2004), por outro lado, são imunes a essas situações, uma vez que eles se baseiam em informações implícitas de diferentes fontes, que podem ser agregadas de forma a gerarem conhecimentos e relações entre produtos. Conceitualmente, a filtragem colaborativa é um processo de *filtragem* de informação a partir de dados provenientes de múltiplas fontes. Cada uma dessas fontes pode ter uma visão global ou parcial do processo que produz os dados, e essas visões podem ser de perspectivas diferentes. Essas múltiplas perspectivas podem então ser utilizadas para corrigir valores, eliminar ruído, completar informações, ou definir tendências, dentre outras aplicações.

Em um sistema de recomendação, essa técnica é utilizada de forma que cada usuário se torna uma das fontes de informação e cada item é parte das informações a serem observadas. Com a observação

parcial dos usuários, é possível então definir grupos de usuários com perfis parecidos e, dentro desses grupos, gerar as recomendações.

Utilizando o exemplo anterior, poderia ser implementado, em uma locadora de filmes, um sistema de avaliação em que cada usuário dará uma nota de 1 a 5 para os filmes já assistidos por eles. Imaginando um cenário em que todos os clientes assistiram a todos os filmes, e atribuíram notas a eles, podemos ver o formato dessa base de dados, gerada colaborativamente, na Tab. 2.9.

Tab. 2.9: Notas que clientes de uma locadora de filmes atribuíram aos filmes já assistidos por eles.

	Filmes									
	Ação						*Comédia*			
Clientes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A	5	5	4	5	4	4	4	3	5	4
B	5	5	5	5	5	5	2	1	1	2
C	5	5	5	5	5	5	1	2	2	3
D	5	5	5	5	5	5	1	3	2	2
E	1	1	2	2	1	1	5	5	5	5
F	2	1	1	1	2	2	5	5	4	5
G	2	1	2	1	2	1	5	4	5	5

Nessa tabela, é possível perceber a formação de grupos bem definidos. O primeiro grupo, composto pelos clientes *A*, *B*, *C* e *D*, tem preferência pelos filmes de ação, enquanto o grupo composto pelos clientes *E*, *F* e *G* expressa preferência pelos filmes de comédia. Note que o cliente *A* poderia estar também no segundo grupo. Porém, as notas atribuídas aos filmes de ação foram mais favoráveis. Como a situação em que todos os clientes assistiram a todos os filmes de uma locadora é improvável, essa tabela tende a ser incompleta, conforme apresentada na Tab. 2.10.

Essa tabela reflete uma situação mais próxima ao que teríamos em uma situação real em sistemas de recomendação. A partir dela, o processo de filtragem colaborativa definiria grupos de usuários, conforme comentado acima e, separando o gosto de cada grupo, faria recomendações. No exemplo acima, o grupo de filmes de ação seria composto pelos clientes *B*, *C* e *D* enquanto o grupo de filmes de comédia seria composto pelos clientes *A*, *E*, *F* e *G*. Definidos os grupos, para cada valor faltante, o filme seria recomendado ao cliente se a nota dele fosse alta para os clientes do mesmo grupo. É possível também observar que a ausência de valores nas notas atribuídas pelo cliente *A* fez com que ele fosse classificado em outro grupo. Note que a atribuição do grupo ao cliente *A* seria melhor resolvida ao aplicar uma técnica que permitisse cada usuário pertencer a múltiplos grupos.

Referente aos conceitos de dados faltantes, esse problema tem uma característica interessante: ele pode conter os três motivos de ausência de dados. Exemplificando com a locadora de filmes, um

Tab. 2.10: Notas que clientes de uma locadora de filmes atribuíram aos filmes já assistidos por eles. Os campos marcados com '-' indicam filmes ainda não assistidos pelos correspondentes clientes, ou assistidos mas sem avaliação.

	Filmes									
	Ação						*Comédia*			
Clientes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A	–	–	4	–	4	4	4	–	5	4
B	5	5	5	5	5	5	–	1	1	–
C	5	5	5	5	5	5	–	–	2	–
D	5	5	5	5	5	5	–	3	–	2
E	1	–	–	2	1	1	5	5	5	5
F	2	1	–	–	–	–	5	5	4	5
G	–	1	–	1	2	–	5	4	5	5

cliente pode dar nota apenas para os filmes que ele gostou, por ter uma opinião mais forte sobre eles, ocasionando ausência de dados do tipo NMAR (Steck, 2010), em que o motivo da ausência é o valor baixo da nota do filme. Outra situação é ele não alugar filmes de determinado gênero e, portanto, ficando impossibilitado de dar notas para esses filmes, ocasionando o tipo MAR, em que o motivo da ausência está relacionado às notas dadas a outros filmes, que já definem sua preferência. Finalmente, o cliente pode deixar de dar nota a um filme por motivos diversos, como falta de tempo, esquecimento, falha no sistema, o que é classificado como um motivo MCAR. Isso dificulta a concepção de técnicas para imputação de tais dados, uma vez que seria necessário identificar o motivo de cada uma das notas estarem ausentes e fazer a análise correspondente. Por conta disso, a grande maioria das técnicas empregadas em filtragem colaborativa (Herlocker et al., 2004; Lemire, 2005; Schafer et al., 2007; de Castro et al., 2007a,c; Candillier et al., 2008) assume apenas uma distribuição MCAR dos dados faltantes.

2.7 Síntese do capítulo

Este capítulo mostrou que, apesar de a tecnologia disponível permitir a aquisição de dados, em grande quantidade, junto a processos de interesse, os dados adquiridos podem conter inúmeras incertezas, como ruído proveniente de interferências internas ou externas ao processo de aquisição, incapacidade em adquirir parte dos dados, dentre outros fatores. Essas incertezas influenciam o processo de análise dos dados, seja no cálculo de índices estatísticos ou na determinação e classificação dos objetos da base em grupos distintos.

A ausência dos dados em um atributo foi então detalhada através da descrição e classificação dos motivos da ausência dos mesmos, seja independente (MCAR), dependente de outros atributos (MAR) ou dependente de seu próprio valor (NMAR). A compreensão desses motivos é necessária para um melhor entendimento da dificuldade e das limitações das técnicas utilizadas no tratamento da ausência de dados.

Algumas das técnicas mais comuns utilizadas para tratar bases de dados incompletas foram apresentadas, incluindo comentários sobre as principais vantagens e desvantagens de cada uma delas. Essas técnicas foram então ilustradas por meio de um exemplo didático, efetuado em uma base de dados gerada artificialmente e que servirá de exemplo em outros capítulos deste trabalho, a qual está devidamente descrita no Apêndice A.

Adicionalmente, foi apresentado (sem ser resolvido aqui) um problema da literatura, com aplicações reais imediatas, denominado filtragem colaborativa, que faz uso de técnicas de imputação de forma a determinar valores ausentes na base definida por esse problema. Foi mostrado também que os três motivos de ausência de dados podem ser encontrados na base utilizada nesse problema, fazendo com que essas bases de dados sejam desafiadoras para métodos tradicionais.

No próximo capítulo, será mostrada a técnica chamada biclusterização, que é capaz de extrair, de uma base de dados, informações contidas em subconjuntos de objetos e atributos. Será mostrado que essa propriedade faz com que essa técnica tenha a capacidade de separar e identificar os casos de ausência de dados, além de eliminar ruído inerente à base de dados.

Capítulo 3

Biclusterização

Técnicas tradicionais de *clusterização* ou *agrupamento*, também conhecido pelo termo em inglês *clustering*, são populares na área de mineração de dados e têm por objetivo organizar e extrair conhecimentos de uma base de dados. O motivo de sua popularidade se deve, inicialmente, ao crescente aumento da capacidade de adquirir e armazenar dados. Grandes quantidades de dados levam à necessidade de ferramentas que automatizem o processo de análise, e o agrupamento é uma das técnicas que podem ser aplicadas para esse fim.

Levando em consideração um conjunto de dados, onde cada elemento é definido como objeto, e cada objeto é representado por um vetor de atributos, as ferramentas de agrupamento, essencialmente, criam subconjuntos de objetos pertencentes a essa base de dados, em que os elementos de um mesmo subconjunto possuam alguma relação de similaridade definida de acordo com o problema a ser tratado.

Para exemplificar, utilizando como base os problemas de imputação de dados faltantes descritos no Cap. 2, um uso comum do agrupamento seria encontrar grupos de objetos que sejam altamente correlacionados e imputar os dados tendo por base apenas esses grupos. Fazendo isso, o espaço dos dados utilizados no momento da imputação é reduzido, de forma que apenas os objetos mais pertinentes aos valores a serem imputados farão parte dos cálculos da estimação desses dados faltantes (Candillier et al., 2008).

Embora essas ferramentas tenham um longo histórico de casos de sucesso em suas aplicações (Jain et al., 1999; Berkhin, 2006), elas possuem algumas limitações. Primeiramente, essas técnicas não são capazes de calcular uma correlação parcial nativamente, isto é, detectar o subconjunto de atributos que melhor aproximem determinado conjunto de objetos. Na literatura, tenta-se suprir essa limitação com as técnicas de *seleção de atributos*, que avaliam os atributos globalmente e tentam manter apenas aqueles que são mais significativos para a base de dados como um todo. Mas isso não é suficiente quando temos situações em que, para cada conjunto de objetos, um determinado

subconjunto de atributos é o mais significativo.

Outra limitação é que, na maioria das técnicas de agrupamento, um mesmo objeto pode pertencer apenas a um único grupo. Essa limitação é uma implicação direta da seleção de atributos, pois um mesmo objeto poderia pertencer a grupos diferentes a partir de conjuntos distintos de atributos. Isso é útil quando é possível extrair informações diferentes de cada objeto, dependendo do conjunto de atributos estudado. Algumas técnicas tentam suprir essa limitação, atribuindo um mesmo objeto a grupos diferentes, caso ele tenha um grau de similaridade considerado alto em mais de um grupo, e associando a relação objeto-grupo com um *grau de pertinência* (Bezdek et al., 1984).

O exemplo didático utilizado no Cap. 2, ilustrado na Tab. 2.9 e reproduzido aqui pela Tab. 3.1, pode ser utilizado para exemplificar os benefícios obtidos em superar tais limitações. Essa tabela, conforme descrito anteriormente, contém as notas que clientes de uma locadora de filmes atribuíram aos filmes assistidos por eles. Em uma abordagem padrão de agrupamento, primeiramente seriam calculadas as similaridades (ou dissimilaridades) entre cada par de clientes utilizando uma métrica de distância ou correlação. Para manter a simplicidade, a distância euclidiana será utilizada como uma medida de dissimilaridade, ou seja, quanto menor o valor, mais similar é um usuário do outro. Os resultados do cálculo de dissimilaridades estão ilustrados na Tab. 3.2.

Tab. 3.1: Notas que clientes de uma loja de filmes atribuíram aos filmes já assistidos por eles.

	Filmes									
	Ação						*Comédia*			
Clientes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A	5	5	4	5	4	4	4	3	5	4
B	5	5	5	5	5	5	2	1	1	2
C	5	5	5	5	5	5	1	2	2	3
D	5	5	5	5	5	5	1	3	2	2
E	1	1	2	2	1	1	5	5	5	5
F	2	1	1	1	2	2	5	5	4	5
G	2	1	2	1	2	1	5	4	5	5

Nessa tabela, é possível verificar a existência de dois grupos distintos de clientes. O primeiro grupo contém clientes que atribuíram notas altas para filmes de ação, representado com a cor cinza claro; o segundo grupo contém os clientes que atribuíram notas altas para filmes de comédia, demarcados com a cor cinza escuro. Mas, analisando o cliente *A* detalhadamente na Tab. 3.1, é possível perceber que as notas que ele atribuiu aos filmes de comédia são similares às notas dadas pelos clientes do grupo cinza escuro. Apesar disso, por conta das notas que esse cliente atribuiu aos filmes de

ação, sua distância em relação ao grupo de comédias se tornou mais alta do que em relação ao grupo de ação.

Tab. 3.2: Distância euclidiana entre os clientes da Tab. 3.1, calculada a partir das notas atribuídas por cada cliente.

Clientes	A	B	C	D	E	F	G
A	**0,00**	**5,57**	**4,79**	**5,00**	8,31	8,06	7,81
B	**5,57**	**0,00**	**2,00**	**2,45**	11,49	10,86	10,86
C	**4,79**	**2,00**	**0,00**	**1,41**	10,95	10,39	10,39
D	**5,00**	**2,45**	**1,41**	**0,00**	10,95	10,39	10,49
E	8,31	11,49	10,95	10,95	**0,00**	**2,45**	**2,00**
F	8,06	10,86	10,39	10,39	**2,45**	**0,00**	**2,00**
G	7,81	10,86	10,39	10,49	**2,00**	**2,00**	**0,00**

Uma vez que, na análise visual, se torna claro que o cliente *A* deveria pertencer também ao grupo de filmes de comédia, é possível separar a lista de atributos em duas listas, nesse caso, uma separação por gênero de filmes. A partir dessa separação, é possível construir duas tabelas de cálculo de dissimilaridades, uma referente aos filmes de ação (Tab. 3.3) e outra aos filmes de comédia (Tab. 3.4).

Os grupos formados, seguindo a similaridade em filmes de ação da Tab. 3.3, são os mesmos da tabela original (Tab. 3.2). Porém, os valores das dissimilaridades obtidas se tornaram menores entre os clientes do mesmo grupo, enquanto entre clientes de grupos distintos a diferença permaneceu elevada.

A Tab. 3.4 apresenta os resultados da distância euclidiana entre os clientes levando em conta apenas os filmes de comédia. Novamente, dois grupos foram formados, mas com o cliente *A* participando do grupo de filmes de comédia.

Tab. 3.3: Distância euclidiana entre os clientes da Tab. 3.1, calculada a partir das notas atribuídas por cada cliente para os filmes de ação.

Clientes	A	B	C	D	E	F	G
A	**0,00**	**1,73**	**1,73**	**1,73**	7,94	7,62	7,61
B	**1,73**	**0,00**	**0,00**	**0,00**	9,05	8,66	8,66
C	**1,73**	**0,00**	**0,00**	**0,00**	9,05	8,66	8,66
D	**1,73**	**0,00**	**0,00**	**0,00**	9,05	8,66	8,66
E	7,94	9,05	9,05	9,05	**0,00**	**2,24**	**1,73**
F	7,61	8,66	8,66	8,66	**2,24**	**0,00**	**1,41**
G	7,61	8,66	8,66	8,66	**1,73**	**1,41**	**0,00**

Tab. 3.4: Distância euclidiana entre os clientes da Tab. 3.1, calculada a partir das notas atribuídas por cada cliente para os filmes de comédia.

Clientes	A	B	C	D	E	F	G
A	0,00	5,29	4,47	4,69	**2,45**	**2,64**	**1,73**
B	5,29	**0,00**	**2,00**	**2,45**	7,07	6,56	6,56
C	4,47	**2,00**	**0,00**	**1,41**	6,16	5,74	5,74
D	4,69	**2,45**	**1,41**	**0,00**	6,16	5,74	5,92
E	**2,45**	7,07	6,16	6,16	**0,00**	**1,00**	**1,00**
F	**2,64**	6,56	5,74	5,74	**1,00**	**0,00**	**1,41**
G	**1,73**	6,56	5,74	5,92	**1,00**	**1,41**	**0,00**

Com as informações obtidas através das Tabelas 3.3 e 3.4, é possível atribuir o cliente *A* a dois grupos distintos e, dessa forma, aumentar a diversidade e acertos nas recomendações de novos filmes. Vale notar que essa separação foi possível apenas pela facilidade de uma análise visual da Tab. 3.1, uma vez que ela contém poucos dados e, assim, as notas atribuídas pelos clientes estão visualmente bem definidas entre os dois gêneros de filmes estudados. Em uma situação real, essa separação visual possivelmente se torne irrealizável, pois ela pode ser proveniente de diversos subconjuntos diferentes de atributos. No caso dos filmes, um grupo de usuários pode ser definido por atributos explícitos como gêneros, atores que participam do filme, diretores, ou até mesmo por similaridades implícitas entre filmes que não compartilham atributos em comum, porém têm um efeito similar sobre o gosto de um grupo de usuários.

Dessa forma, a tarefa de separar subconjuntos de atributos para encontrar grupos distintos foge de uma análise simplificada ou de uma técnica de seleção de atributos, uma vez que não basta eliminar alguns atributos ou unificá-los em um atributo único. Para esse objetivo, a técnica de agrupamento deve ir além e ser capaz de agrupar o conjunto de objetos e o conjunto de atributos simultaneamente. Uma técnica que possui tal característica é denominada *biclusterização* (Madeira & Oliveira, 2004; de França et al., 2006).

Este capítulo introduzirá o conceito de *biclusterização*, capaz de agrupar os objetos de uma base de dados juntamente com seus atributos, tornando possível relacionar os objetos condicionalmente aos atributos utilizados em um dado agrupamento. Conforme será mostrado, existem diversas formulações possíveis para o problema de biclusterização, cada qual com suas propriedades e utilidades. Em seguida, o capítulo focará em uma dessas formulações, denominada *δ–bicluster*, que será a formulação utilizada no restante desta tese. Para concluir, várias técnicas de biclusterização, heurísticas e meta-heurísticas, serão apresentadas resumidamente para ilustrar as formas possíveis de realização desta tarefa.

3.1 Definições do problema de biclusterização

Conforme dito anteriormente, a biclusterização, do inglês *biclustering*, também chamada de *co-clustering*, é a denominação dada às técnicas de agrupamento que buscam atuar junto a uma base de dados agrupando simultaneamente os objetos e os atributos (Madeira & Oliveira, 2004; de França et al., 2006). Como comparação com as técnicas clássicas de agrupamento, a biclusterização pode ser vista como um *modelo local* de agrupamento, levando em conta que ela considera apenas parte dos atributos de cada objeto para definir os grupos. Como contraste, as técnicas clássicas de agrupamento podem ser definidas como pertencentes a um *modelo global*, pois levam em conta todos os atributos no momento de definir os grupos. Um exemplo dessas duas definições pode ser visto na Fig. 3.1.

$$\begin{bmatrix} 5 & 5 & 4 & 5 & 4 & 4 & 4 & 3 & 5 & 4 & 1 \\ 5 & 5 & 5 & 5 & 5 & 5 & 2 & 1 & 1 & 2 & 2 \\ 5 & 5 & 5 & 5 & 5 & 5 & 1 & 2 & 2 & 3 & 3 \\ 5 & 5 & 5 & 5 & 5 & 5 & 1 & 3 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 5 & 5 & 5 & 5 & 2 \\ 2 & 1 & 1 & 1 & 2 & 2 & 5 & 5 & 4 & 5 & 1 \\ 2 & 1 & 2 & 1 & 2 & 1 & 5 & 4 & 5 & 5 & 1 \end{bmatrix}$$

Fig. 3.1: Base de dados do exemplo ilustrado na Tab. 3.1 com um possível grupo do modelo global, demarcado com um tom claro, e um grupo de modelo local em tom escuro.

Nesta figura, lembrando o exemplo dado na seção anterior, é possível notar que o primeiro objeto da base de dados não pertenceria ao mesmo grupo dos últimos objetos em um modelo global. Mas, no modelo local, utilizando técnicas de biclusterização, ele pode pertencer aos dois grupos marcados na figura, aumentando a flexibilidade de informações que podem ser extraídas da base. Vale notar que as soluções obtidas por um modelo global podem estar inclusas nas soluções da biclusterização e, portanto, o conjunto de soluções do modelo global não é exclusivo desse modelo.

Cabe salientar aqui que o agrupamento simultâneo de objetos e atributos, executado pela biclusterização, não é equivalente à execução sequencial ou até cíclica de duas clusterizações, uma aplicada aos objetos e a outra aos atributos. Apesar da flexibilidade obtida pelo uso de biclusterização, em detrimento a ferramentas de agrupamento, essa técnica deve ser utilizada apenas quando suas características forem necessárias, pois o espaço de busca formado por ela tende a ser significativamente maior do que na formulação de um modelo global. Situações em que o uso de biclusterização é pertinente incluem: existência de padrões que são encontrados apenas quando utilizadas partes dos atributos; possibilidade de um objeto participar em mais de um grupo, dependendo dos atributos utilizados; e existência de apenas parte dos objetos contendo informações relevantes, dado um conjunto dos atributos.

Na literatura, não existe um consenso em torno das restrições em relação às características de um bicluster. Uma delas, relativa à unicidade de um grupo em relação aos objetos ou atributos, aponta que um determinado grupo de objetos pode ser definido somente em relação a um grupo de atributos, e vice-versa. Porém, isso implica que a análise feita em um subconjunto de atributos esteja limitada apenas a um grupo único de objetos, o que nem sempre é verdade e pode ser verificado nas Tabelas 3.3 e 3.4, onde, para um dado subconjunto de atributos, existem dois grupos distintos de objetos.

Outra restrição diz respeito à exclusividade de objetos e atributos dentro de um bicluster. Alguns algoritmos (Kluger et al., 2003) limitam um objeto ou atributo a pertencer a apenas um bicluster. Essa característica geralmente é denominada *biclusters sem sobreposição*. Já outros algoritmos (Cheng & Church, 2000) suportam que um objeto ou atributo possa pertencer a mais de um bicluster, desde que dois biclusters respeitem uma taxa máxima de sobreposição; esses biclusters são chamados de *biclusters com restrição de sobreposição*.

Além disso, em muitas situações, a medida de similaridade entre os objetos utilizada pelo método de biclusterização deve ser robusto em relação ao ruído. Isso porque, ao reduzir a dimensão dos objetos, tomando apenas um subconjunto de atributos, o bicluster fica sujeito a uma menor diluição do ruído que poderia ocorrer em uma análise com todos os elementos da base de dados. Dessa forma o método deve ser capaz de tratar o ruído e auto-ajustar o cálculo da similaridade ou o limiar que define um grupo, em relação a cada bicluster. Apesar disso, alguns métodos de biclusterização buscam biclusters *puros*, livres de qualquer ruído; esses métodos geralmente são aplicados em bases de dados binárias.

3.1.1 Definição formal do problema de biclusterização

Esta seção descreve uma definição formal aqui proposta para facilitar o entendimento deste texto. Para tanto, será utilizada uma matriz de dados genérica A, contendo n linhas e m colunas, em que cada elemento é representado por a_{ij}, e $i \in \{1, ..., n\}$ é o índice da linha e $j \in \{1, ..., m\}$ é o índice da coluna. Denomina-se $L = \{1, ..., n\}$ e $C = \{1, ..., m\}$ como o conjunto de linhas e colunas, respectivamente, da matriz A, ou seja, a matriz A pode ser descrita como $A(L, C)$.

Consequentemente, se tivermos $I \subseteq L$ e $J \subseteq C$ como subconjuntos das linhas e colunas, respectivamente, denota-se $A(I, J)$ como a sub-matriz de A contendo apenas os elementos a_{ij} da matriz que tenham índices dentro dos conjuntos I e J.

Por definição, um *agrupamento de linhas ou objetos* é uma submatriz de A que contém uma determinada similaridade entre suas linhas perante todos os atributos. Então, um agrupamento de linhas pode ser descrito como $A(I, C)$, onde $I = \{i_1, ..., i_k\}$ é um subconjunto contendo $k \leq n$ linhas, com $i_l \in L$ e $l \in \{1, ..., k\}$, e C é o conjunto de todas as colunas.

Analogamente, um *agrupamento de colunas ou atributos* é uma sub-matriz de A definida por $A(L, J)$, com $J = \{j_1, ..., j_s\}$, sendo $s \leq m$, e tal que $j_c \in C$ e $c \in \{1, ..., s\}$, e L é o conjunto contendo todas as linhas, sendo que os elementos dessa sub-matriz possuem similaridades entre si.

Partindo das definições acima, um *bicluster* pode ser definido como a submatriz de A definida por $A(I, J)$, com $I = \{i_1, ..., i_k\}$ e $J = \{j_1, ..., j_s\}$, sendo $k \leq n$ e $s \leq m$, cujos elementos apresentam alguma similaridade definida pelo problema estudado. Um bicluster $A(I, J)$ pode, então, ser definido como uma sub-matriz $k \times s$ formada por linhas e colunas arbitrárias de A. Dada a dimensão de um bicluster, define-se também o termo *volume*, representado por V e denotando o tamanho do espaço coberto por ele, sendo então $V = k \cdot s$. Apesar desse conceito remeter à ideia de *área*, o termo volume é amplamente utilizado na literatura (Cheng & Church, 2000; Agrawal et al., 1998).

O problema geral de biclusterização pode ser, então, definido como: dada uma matriz A de dimensão $n \times m$, encontrar um conjunto de biclusters $B_r = (I_r, J_r)$ com $r = 1, ..., t$, sendo que cada bicluster B_r satisfaz alguma condição de homogeneidade.

Considerando agora um conjunto de biclusters, torna-se relevante considerar o conceito de *sobreposição*, que indica o quanto um determinado bicluster está contido em outro. Esse conceito pode ser formalizado de acordo com a Eq. 3.1, dados os biclusters B_k e B_l:

$$\text{Sobreposição}(k, l) = \frac{|I_k \cap I_l| \cdot |J_k \cap J_l|}{|I_l| \cdot |J_l|}. \tag{3.1}$$

Outro conceito associado ao conjunto de biclusters gerado é a *cobertura*, que representa a porcentagem dos elementos da base de dados contidos no conjunto de biclusters. Esse conceito pode ser formalizado de acordo com a Eq. 3.2, considerando l biclusters gerados:

$$\text{Cobertura}(l) = \frac{|I_1 \cup I_2 ... \cup I_l| \cdot |J_1 \cup J_2 ... \cup ...J_l|}{n \cdot m}. \tag{3.2}$$

Conforme é possível perceber, o problema de biclusterização é genérico o suficiente para permitir uma flexibilização no critério de geração e nas características dos biclusters encontrados. Diante disso, a complexidade do problema de biclusterização depende diretamente de como este é abordado. Se levarmos em conta a sua forma mais simples, quando A é uma matriz binária, um bicluster corresponde a um biclique de um grafo bipartido, supondo que as linhas da matriz A são os vértices à esquerda e as colunas os vértices à direita, e os valores dos elementos indicam se existe uma aresta ligando uma linha a uma coluna, caso o valor seja 1, ou 0, caso contrário.

Encontrar o bicluster de tamanho máximo, nessa situação, é equivalente a encontrar o biclique com o máximo de arestas em um grafo bipartido, que é um problema NP-completo (Peeters, 2003). Consequentemente, em casos mais complexos, é possível inferir que estes também sejam NP-completos. Diante disso, boa parte dos algoritmos de biclusterização são métodos inexatos, baseados

em heurísticas ou meta-heurísticas.

3.2 Considerações ao formular um problema de biclusterização

Dada a flexibilidade em extração de dados e conhecimentos que as técnicas de biclusterização trazem em relação às de clusterização, diversas aplicações para essa técnica foram vislumbradas. Algumas delas já eram tratadas via clusterização, como mineração de texto (Dhillon, 2001; Feldman & Sanger, 2006; de Castro et al., 2007b; de Castro et al., 2010) e filtragem colaborativa (de Castro et al., 2007a,c; Symeonidis et al., 2007; de França et al., 2009), outras são baseadas em problemas de clusterização, porém com objetivos diferentes, como redução de dimensionalidade (Agrawal et al., 1998) e análise de dados de expressão gênica (Agrawal et al., 1998; Tang et al., 2001; Mitra & Banka, 2006; Mitra et al., 2006; Divina & Aguilar–Ruiz, 2007; Giráldez et al., 2007; Maulik et al., 2008).

Em relação às aplicações de mineração de textos, em Dhillon (2001), o autor formulou o problema de biclusterização como um problema de particionamento de grafos bipartidos. Essa formulação é possível ao separarmos os conjuntos de vértices como $V_1 = L$ e $V_2 = C$, e o conjunto de arestas $E = e_{ij}$ sendo representado por

$$e_{ij} = \begin{cases} 1, & \text{se } a_{ij} \neq 0 \\ 0, & \text{c.c.} \end{cases}, \tag{3.3}$$

com a_{ij} representando a frequência com que a palavra j ocorre no documento i.

Dessa forma, o autor definiu o problema de biclusterização como aquele de encontrar a partição de vértices com corte mínimo em um grafo bipartido. Esse problema tem como característica a exclusividade de vértices em cada agrupamento, ou seja, um objeto ou atributo não pode pertencer a mais de um bicluster. Apesar dessa limitação, essa formulação facilita a busca pela solução do problema, pois permite que diversas técnicas de clusterização possam ser utilizadas para isso.

Outra aplicação para mineração de textos foi feita em de Castro et al. (2007b), em que uma meta-heurística imuno-inspirada foi utilizada para encontrar os biclusters dentro de uma base de dados A, sendo que cada elemento a_{ij} representava a frequência com a palavra j aparecia no documento i. Na formulação para esse problema, os autores criaram uma combinação linear de funções para atender dois objetivos distintos: maximizar o volume do bicluster e minimizar o erro quadrático médio, denominado resíduo, entre o bicluster e o seu modelo ideal. O modelo ideal é a representação do bicluster se ele apresentasse uma correlação perfeita entre seus elementos. A intenção dessa função-objetivo era encontrar biclusters de volume máximo cujos elementos apresentassem uma correlação positiva alta, levando à extração de termos que definiam uma classe de documentos. Esse método também permitia que uma palavra ou um documento pertencesse a mais de um bicluster, tornando possível

encontrar sub-classificações dos documentos da base.

Dando sequência a esse estudo, em de Castro et al. (2010), os autores aplicaram esse mesmo algoritmo de biclusterização para a *Expansão de Termos de Pesquisa*, em que, dado um termo de pesquisa em uma base de busca de documentos, o algoritmo procura termos relacionados a esse que podem tornar os resultados de busca mais completos e/ou específicos. O uso de biclusterização nessa aplicação é pertinente, pois um mesmo termo de busca pode estar vinculado a contextos diferentes. Portanto, deve ser expandido para conjuntos de termos que caracterizam agrupamentos distintos.

Um problema comum, enfrentado em muitas aplicações de mineração de dados, é quando a base de dados a ser analisada apresenta um número de atributos elevado e que introduzem ruído na análise, em vez de acrescentar informações. Isso geralmente ocorre pois, durante a coleta de dados, o analista não tem um conhecimento prévio de quais atributos são determinantes para classificar um objeto. Por conta disso, é necessário adquirir o máximo de informação possível sobre os objetos de forma a não correr o risco de uma análise incompleta ou polarizada. Porém, alguns atributos podem distanciar um objeto de seu grupo, fazendo com que ele pareça pertencer a outro grupo diferente. Um exemplo disso foi visto anteriormente na Tab. 3.1, onde a mistura de gêneros diferentes de filmes pode mudar o grupo a que um objeto é atribuído. Além disso, alguns atributos podem ter ausência de significado, ou seja, possuírem valores aleatórios e sem qualquer relação com os agrupamentos estudados. Isso pode gerar ruído na análise de dados ou até mesmo levar a conclusões errôneas ao final da análise.

Para solucionar esse problema, geralmente é aplicada uma técnica de *redução de dimensionalidade*, que consiste em transformar os atributos originais de uma base de dados em um espaço reduzido de atributos, que geralmente são combinações lineares dos atributos da base original. Apesar de essa técnica solucionar parcialmente o problema, em Agrawal et al. (1998), os autores identificaram algumas desvantagens no uso dessa metodologia. Primeiramente, os atributos são agrupados e transformados de forma única, ou seja, cada atributo aparece em apenas uma combinação linear para gerar um novo atributo. Consequentemente, isso inibe a capacidade de *interpretação* dos agrupamentos encontrados, uma vez que os novos atributos não possuem uma descrição bem definida. Nesse artigo, os autores propuseram uma técnica de biclusterização que procura por regiões inter-relacionadas, na base de dados, seguindo algum critério de agrupamento, partindo de um único elemento e expandindo até encontrar um agrupamento máximo.

Uma aplicação na área de análise de dados biológicos, e a que mais utiliza técnicas de biclusterização, é o estudo de *expressão gênica*. Esse estudo busca a compreensão de como os genes são regulados sob certas condições. A análise desses dados se tornou possível com a criação das técnicas de *microarranjo de DNA* (Lazzeroni & Owen, 2002), que basicamente consiste em tomar amostras de genes a serem comparadas, cada qual em diferentes circunstâncias (ex., células saudáveis e células doentes), e utilizar substâncias que *marcam* os genes em cores diferentes caso estes estejam

sobre-expressos ou sub-expressos. Com essa técnica, é possível gerar uma base de dados contendo os níveis de expressão de múltiplos genes (chegando à ordem de milhares) sob múltiplas condições experimentais (chegando à ordem de centenas).

Entretanto, na maioria das situações, apenas uma pequena parcela dos genes participa do processo celular de interesse no estudo realizado. E, dentre esses genes, apenas uma pequena parcela das condições experimentais estão ativas. Além disso, um gene pode participar de vários grupos distintos, que podem ou não estarem *coativos* sob as diversas condições experimentais. Diante dessas características, a área de estudo de expressões gênicas se tornou a mais ativa na criação de métodos de biclusterização, sendo responsável por grande parte dos algoritmos criados para esse fim. Muitos dos métodos criados para esse estudo serão descritos nas próximas seções e, por esse motivo, não serão abordados aqui.

Outra aplicação de biclusterização, embora ainda com poucas publicações, são as técnicas de filtragem colaborativa, descritas no Cap. 2. Algumas das propriedades que tornam o uso de biclusterização interessante para essa técnica foram ilustradas no exemplo dado no início deste capítulo pela Tab. 3.1. O motivo para o baixo índice de publicações nessa área, até o presente momento, é a necessidade de criar mecanismos que consigam lidar com os dados faltantes das bases de dados de filtragem colaborativa, que geralmente são altamente esparsas. Uma forma de tratar esse problema foi exposta em de Castro et al. (2007a,c), em que os valores faltantes foram substituídos por dados numéricos com valor zero e, após a biclusterização, os dados faltantes eram imputados através de uma média ponderada entre os valores contidos no bicluster.

Em Symeonidis et al. (2007), a base de dados de filtragem colaborativa foi convertida para uma base binária, onde o valor 0 representa se um cliente avaliou negativamente ou não avaliou o produto e 1 caso tenha avaliado positivamente. Com a base transformada para a forma binária, os autores aplicaram uma técnica de biclusterização que procura por biclusters com valores constantes, nesse caso, biclusters cujos valores são todos iguais a 1. A estratégia adotada foi uma metodologia similar à *dividir-e-conquistar* capaz de encontrar todos os biclusters máximos dentro da base, a partir dessa formulação. Após os biclusters terem sido gerados, o algoritmo aplica um método de k-vizinhos mais próximos para encontrar os biclusters mais próximos a cada usuário e, então, usá-los para gerar as recomendações.

Uma terceira abordagem foi proposta em de França et al. (2009), em que um algoritmo multi-objetivo foi aplicado para encontrar biclusters. Nesse algoritmo, os objetivos principais foram a maximização do volume e da correlação dos elementos do bicluster. Além disso, outros objetivos foram formulados em forma de restrição, como a taxa mínima de elementos não-nulos dentro do bicluster, a taxa de sobreposição entre eles e o limite mínimo da correlação média entre os elementos. Após os biclusters terem sido encontrados, o algoritmo passa a encarar os valores faltantes como

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Fig. 3.2: Bicluster com valores *perfeitamente* constantes tanto nas linhas quanto nas colunas, com $\mu = 1$.

variáveis de um problema de otimização quadrática, obtendo então uma solução ótima da predição nas condições de correlação dos elementos de cada bicluster. Essa metodologia de imputação de dados faltantes será devidamente formalizada no Cap. 5.

Na próxima seção, as principais medidas de qualidade de um bicluster serão apresentadas e detalhadas, assim como algumas terminologias que serão utilizadas no restante desta tese.

3.3 Tipos de bicluster

Um primeiro passo para a formulação do problema de biclusterização que um algoritmo vai tratar é definir o critério de avaliação da qualidade do bicluster (Madeira & Oliveira, 2004). O critério de avaliação define o tipo de correlação que os valores dos elementos dentro de um bicluster devem ter. Os tipos mais comuns de critério de avaliação são:

- Biclusters com valores constantes.
- Biclusters com valores constantes nas linhas ou nas colunas.
- Biclusters com valores coerentes.
- Biclusters com evoluções coerentes.

A seguir, cada um desses critérios de avaliação será explicado mais detalhadamente.

3.3.1 Biclusters com valores constantes

Um bicluster com os valores de elementos constantes é definido por

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu, (i, j) \in (I, J) \right\}, \tag{3.4}$$

onde μ é um valor constante, e exemplificado na Fig. 3.2.

Esse critério de avaliação é o mais simples e procura por biclusters em que os objetos se comportam de maneira igual dentro do subconjunto de atributos. Para o caso de filtragem colaborativa, essa forma de avaliação procuraria por biclusters em que os clientes atribuem a mesma nota aos itens do subconjunto. Apesar de ser uma avaliação simples, esse tipo de formulação requer que a base de dados não apresente ruído para ter sucesso na busca de biclusters perfeitos, ou então que ela passe por um pré-processamento, conforme feito em Symeonidis et al. (2007), que transformou a base de dados em uma matriz binária para filtragem colaborativa.

Mas, em alguns casos, não é possível encontrar biclusters perfeitamente constantes, e é necessário admitir pequenas variações ou ruídos em seus valores. Isso pode ser feito incluindo um intervalo de valores aceitáveis na função-objetivo, avaliando a variância dos valores do bicluster, ou qualquer outra função-objetivo robusta ao ruído. Na subseção 3.4.1, será detalhado um algoritmo que usa a variância como função-objetivo, denominado *Block Clustering* (Hartigan, 1972).

3.3.2 Biclusters com valores constantes nas linhas ou nas colunas

Num tipo de avaliação similar ao proposto na subseção 3.3.1, os biclusters com valores constantes nas linhas ou nas colunas são biclusters em que os valores são constantes apenas em uma das dimensões. Eles são representados por um valor constante que serve como *base* e cada linha ou coluna possui um termo aditivo ou multiplicativo em relação a esse valor-base. Então, representa-se um bicluster *perfeito* com valores constantes nas linhas como segue:

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu + \alpha_i, (i, j) \in (I, J) \right\}, \tag{3.5}$$

ou

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu \times \alpha_i, (i, j) \in (I, J) \right\}, \tag{3.6}$$

onde α_i representa o ajuste para o valor da linha i em relação ao valor base μ, podendo ser um ajuste aditivo ou multiplicativo.

Analogamente, para um bicluster *perfeito* com valores constantes nas colunas, representa-se como:

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu + \beta_j, (i, j) \in (I, J) \right\}, \tag{3.7}$$

ou

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu \times \beta_j, (i, j) \in (I, J) \right\}, \tag{3.8}$$

onde β_j representa o ajuste para o valor da coluna j em relação ao valor base μ, podendo ser um ajuste aditivo ou multiplicativo.

Exemplos desse tipo de bicluster podem ser encontrados nas Figs. 3.3 e 3.4.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 5 & 5 & 5 \\ 0 & 0 & 0 \end{bmatrix}$$

Fig. 3.3: Bicluster com valores perfeitamente constantes nas linhas. O valor base desse bicluster pode ser tomado como $\mu = 1$ e o vetor de ajuste aditivo assume os valores $\alpha = \{0; 1; 4; -1\}$.

$$\begin{bmatrix} 2 & 6 & 1 \\ 2 & 6 & 1 \\ 2 & 6 & 1 \\ 2 & 6 & 1 \end{bmatrix}$$

Fig. 3.4: Bicluster com valores perfeitamente constantes nas colunas. O valor base desse bicluster pode ser tomado como $\mu = 2$ e o vetor de ajuste multiplicativo assume os valores $\beta = \{1; 3; 0, 5\}$.

Para a aplicação de filtragem colaborativa, esse tipo de bicluster encontraria usuários que atribuem as mesmas notas ao subconjunto de itens (constantes nas colunas) ou usuários que seguem uma mesma tendência na atribuição de notas, mas cada qual com seu viés (constantes nas linhas).

Como no caso anterior, o ruído inerente em bases de dados reais pode tornar impossível encontrar biclusters com valores perfeitamente constantes. Mas, nesse caso, pelo fato de o valor variar em função da linha ou da coluna, não é possível avaliar a variância dos valores dentro do bicluster. Uma forma de abordar esse problema é normalizando as linhas ou as colunas usando as respectivas médias. Dessa forma, os biclusters exemplos das Figs. 3.3 e 3.4 se tornariam completamente constantes (Getz et al., 2000). Outra forma é avaliar a variância dos valores em cada linha ou coluna individualmente e definir um limiar para essa variância dentro do bicluster (Califano et al., 2000).

3.3.3 Biclusters com valores coerentes

Uma forma de avaliação, que atualmente é a mais popular entre algoritmos de biclusterização, é a medida de coerência entre os elementos do bicluster. Essa medida calcula uma forma de covariância entre os valores da submatriz formada, a fim de identificar objetos com comportamentos similares em um subconjunto de atributos ou atributos com comportamentos similares em um subconjunto de objetos. Com essa avaliação, é possível identificar tanto biclusters das formas de avaliação descritas

anteriormente como novos tipos de biclusters, que podem possuir informações relevantes de acordo com a aplicação pretendida. Contextualizando para o problema de filtragem colaborativa, um bicluster coerente poderia agrupar clientes que têm tendências similares nas notas atribuídas a determinados itens, ou seja, um cliente A tende a dar notas altas para os itens que avalia, enquanto o cliente B é mais conservador em suas avaliações, porém ambos os clientes seguem a mesma tendência em um subconjunto de itens. Esse tipo de bicluster é exemplificado pela matriz ilustrada na Fig. 3.5.

$$\begin{bmatrix} 10 & 8 & 7 \\ 7 & 5 & 4 \\ 9 & 7 & 6 \\ 4 & 2 & 1 \end{bmatrix}$$

Fig. 3.5: Bicluster com valores coerentes de forma aditiva. Os valores de cada linha são iguais aos valores da primeira linha acrescidos de $\alpha = \{0; -3; -1; -6\}$ e os valores de cada coluna são iguais aos valores da primeira coluna acrescidos de $\beta = \{0; -2; -3\}$.

Nessa figura, é possível perceber que os elementos de cada linha podem ser representados como os elementos de uma das linhas acrescida de uma lista de valores ou, da mesma forma, os elementos de cada coluna podem ser representados como os elementos de uma das colunas acrescida de uma lista de valores. De forma geral, um bicluster perfeitamente coerente pode então ser representado por

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu + \alpha_i + \beta_j, (i, j) \in (I, J) \right\}, \tag{3.9}$$

no caso aditivo, onde μ seria o valor base do bicluster, α_i o valor de ajuste aditivo da linha i e β_j o ajuste aditivo da coluna j e

$$B = A(I, J) = \left\{ a_{ij} \mid a_{ij} = \mu' \times \alpha'_i \times \beta'_j, (i, j) \in (I, J) \right\}, \tag{3.10}$$

no caso multiplicativo, onde μ' seria o valor-base do bicluster, α'_i o valor de ajuste multiplicativo da linha i e β'_j o valor de ajuste multiplicativo da coluna j.

É possível perceber que sempre é possível expressar a coerência multiplicativa na forma aditiva. Também, levando em consideração a forma aditiva, se for feito $\alpha_i = 0, \forall i \in I$, tem-se um bicluster com linhas constantes. Analogamente, ao se adotar $\beta_j = 0, \forall j \in J$, encontram-se biclusters com colunas constantes e com $\alpha_i = 0$ e $\beta_j = 0$, biclusters completamente constantes. Então, a formulação de coerência aditiva pode ser considerada uma formulação geral para encontrar diversos tipos de biclusters interessantes para a análise de dados.

Da mesma forma que nos outros tipos de avaliação, é improvável encontrar biclusters perfeitos dentro de uma base de dados. A forma de avaliar a coerência de um bicluster deve, então, levar em

conta o ruído encontrado nessa base. O modo mais usual para avaliar a coerência de um bicluster é calcular a diferença entre o bicluster encontrado e o modelo perfeito de coerência mais próximo dele. Cheng & Church (Cheng & Church, 2000) foram os primeiros a utilizar essa forma de avaliação, denominando essa função-objetivo de *Resíduo Quadrático Médio* (RQM).

Dado um bicluster B perfeitamente coerente, os termos de ajuste aditivo α_i e β_j são inicialmente desconhecidos. Porém, é possível reescrever a Eq. 3.9 de forma que um elemento a_{ij} do bicluster possa ser obtido através da média a_{IJ} dos elementos contidos nele, da média dos elementos de cada linha, a_{iJ}, e da média dos elementos de cada coluna, a_{Ij}. Esses valores são dados por:

$$a_{iJ} = \mu + \alpha_i + \overline{\beta}, \tag{3.11}$$

$$a_{Ij} = \mu + \overline{\alpha} + \beta_j, \tag{3.12}$$

$$a_{IJ} = \mu + \overline{\alpha} + \overline{\beta}, \tag{3.13}$$

onde $\overline{\alpha}$ é a média do termo de ajuste aditivo das linhas e $\overline{\beta}$ é a média do termo de ajuste aditivo das colunas.

Isolando α_i, β_j e μ nas Eqs. 3.11 , 3.12 e 3.13 acima, obtém-se:

$$\alpha_i = a_{iJ} - \mu - \overline{\beta}, \tag{3.14}$$

$$\beta_j = a_{Ij} - \mu - \overline{\alpha}, \tag{3.15}$$

$$\mu = a_{IJ} - \overline{\alpha} - \overline{\beta}. \tag{3.16}$$

Substituindo a Eq. 3.16 nas Eqs. 3.14 e 3.15:

$$\alpha_i = a_{iJ} - a_{IJ} + \overline{\alpha}, \tag{3.17}$$

$$\beta_j = a_{Ij} - a_{IJ} + \overline{\beta}. \tag{3.18}$$

E, finalmente, substituindo as Eqs. 3.16, 3.17 e 3.18 na fórmula para a_{ij} presente na Eq. 3.9 chega-se a:

$$a_{ij} = a_{Ij} + a_{iJ} - a_{IJ}. \tag{3.19}$$

Essa expressão torna possível calcular qualquer valor de um bicluster perfeitamente coerente apenas utilizando a média de cada linha, de cada coluna e a média total do bicluster. O RQM pode então

ser obtido calculando a diferença quadrática entre o valor real de um bicluster qualquer, com o valor fornecido pela Eq. 3.19:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2, \tag{3.20}$$

onde $H(I,J)$ é o RQM do bicluster $B = A(I,J)$, $|\zeta|$ expressa a quantidade de elementos no conjunto ζ. Analogamente, pode-se definir o RQM da linha i como sendo

$$H(i,J) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2, \tag{3.21}$$

e o RQM da coluna j na forma

$$H(I,j) = \frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2. \tag{3.22}$$

Essas medidas de avaliação são utilizadas pela maioria dos métodos de biclusterização que buscam biclusters coerentes. Alguns desses algoritmos serão explicados em detalhes em seções posteriores deste capítulo.

3.3.4 Biclusters com evolução coerente

A *evolução coerente* difere da medida de coerência descrita anteriormente por não buscar valores coerentes dentro do bicluster, mas sim propriedades coerentes nos valores. Exemplificando na situação de filtragem colaborativa, um bicluster com evolução coerente poderia ser um bicluster em que todas as avaliações de itens dentro dele são maiores que um limiar (avaliações positivas). Outra forma de definir evolução é considerar estados diferentes que os valores da base de dados podem representar. Novamente, em filtragem colaborativa, quatro estados poderiam ser definidos sugerindo a opinião verbal do cliente em relação a um determinado item: *gostou*, *não gostou*, *neutro*, *não opinou*.

Partindo desse conceito, o bicluster pode ter a evolução coerente em todos os seus elementos, apenas nas linhas ou apenas nas colunas, gerando uma similaridade com o conceito de biclusters com valores constantes. Essas três situações estão representadas nas matrizes ilustradas na Fig. 3.6.

Medidas de qualidade para mensurar a evolução coerente de um bicluster podem ser o tamanho do bicluster, relaxando o quanto de seus elementos são coerentes dado um limiar (Murali & Kasif, 2002), ou então o quanto desse bicluster é coerente (Tanay et al., 2002). Medidas similares às utilizadas nos biclusters de valores constantes podem ser também utilizadas.

$$\begin{bmatrix} S & S & S \\ S & S & S \\ S & S & S \\ S & S & S \end{bmatrix} \quad \begin{bmatrix} S_1 & S_1 & S_1 \\ S_2 & S_2 & S_2 \\ S_3 & S_3 & S_3 \\ S_4 & S_4 & S_4 \end{bmatrix} \quad \begin{bmatrix} S_1 & S_2 & S_3 \\ S_1 & S_2 & S_3 \\ S_1 & S_2 & S_3 \\ S_1 & S_2 & S_3 \end{bmatrix}$$

(a) (b) (c)

Fig. 3.6: Biclusters com evolução coerente, sendo que em (a) a evolução é coerente por todo o bicluster, com S representando o estado dos valores; (b) a evolução é coerente nas linhas, com S_i representando o estado dos valores da linha i; e (c) a evolução é coerente nas colunas, com S_i representando o estado dos valores da coluna i.

3.4 Algoritmos de biclusterização

Nesta seção, será descrito pelo menos um algoritmo clássico para cada uma das medidas de qualidade dos biclusters. Como este trabalho focará em biclusterização com valores coerentes, os algoritmos pertencentes a essa classe terão maior ênfase.

3.4.1 Block Clustering

Historicamente, o algoritmo denominado *Block Clustering* e criado por J. A. Hartigan (Hartigan, 1972) foi o primeiro algoritmo a propor uma formulação de biclusterização, embora esse termo tenha sido cunhado posteriormente por Mirkin (1996). Esse algoritmo foi criado para particionar uma base de dados, horizontalmente e verticalmente, de forma a obter submatrizes com baixa variância nos valores, ou seja, próximo de serem constantes.

A qualidade de cada partição nesse algoritmo é dada por

$$VAR(B_p) = VAR(A(I_p, J_p)) = \sum_{i \in I_p, j \in J_p} (a_{ij} - a_{IJ})^2, \tag{3.23}$$

e a qualidade do resultado total do algoritmo

$$VAR(B_1, B_2, ..., B_p) = \sum_{p=1}^{P} \sum_{i \in I_p, j \in J_p} (a_{ij} - a_{IJ})^2, \tag{3.24}$$

onde P é o parâmetro de entrada do algoritmo, que define o número de partições que serão encontradas.

O objetivo principal desse método é a minimização da Eq. 3.24. O algoritmo inicia com a base de dados não-particionada e verifica qual partição gera a maior redução da variância total dos valores. Note que, se a partição ocorresse apenas nas linhas, o número de partições a serem testadas seria igual a 2^k, lembrando que k é definido como o número de linhas em um bicluster. Porém, o autor percebeu

que a melhor partição horizontal ocorre na ordem da média das linhas da partição atual, ou seja, a divisão da partição B em duas partições B' e B'' se dará de tal forma que as médias das linhas em B' serão todas menores do que as em B''. Isso reduz o espaço de busca para $k - 1$ partições a serem testadas horizontalmente. Analogamente, o mesmo é feito para colunas, levando a $s - 1$ partições testadas, e um total de $k + s - 2$ partições a cada passo do algoritmo. O procedimento é repetido até que as P partições sejam encontradas.

Analisando o algoritmo, é possível perceber que, além de procurar por biclusters próximos de valores constantes, ele não permite a sobreposição de partições, ou seja, um elemento irá pertencer a apenas um único bicluster. O algoritmo foi comparado com uma *clusterização hierárquica* (Johnson, 1967) feita separadamente em relação aos objetos e aos atributos, utilizando uma base de dados eleitoral dos Estados Unidos. Essa base relaciona o número de votos recebidos por um certo partido em cada estado durante os anos de 1900 e 1968.

A conclusão a que o autor chegou foi de que as partições obtidas pelos dois métodos são equivalentes, com poucas diferenças na análise final dos resultados. Apesar disso, o algoritmo de Block Clustering gerou um número menor de partições, excluindo partições sem significado, e com desempenho computacional melhor que o método de clusterização hierárquica.

3.4.2 Coupled Two-Way Clustering

O algoritmo *Coupled Two-Way Clustering* (Getz et al., 2000) foi criado para encontrar submatrizes de uma matriz de expressão gênica que contém relações entre genes e experimentos de difícil identificação, devido ao ruído contido na metodologia de microarranjo. Apesar de ser um método de biclusterização, nesse artigo foi empregado o termo *Two-Way Clustering* para descrever a formulação do problema.

Para o funcionamento desse algoritmo, inicialmente é necessário fazer uma normalização da base original de dados. Essa normalização é obtida primeiramente dividindo cada coluna por sua média. Portanto:

$$A' = a'_{ij} = \frac{a_{ij}}{a_{Ij}}. \tag{3.25}$$

Em seguida, normaliza-se cada linha de tal forma a obter valores com média nula e norma unitária:

$$A'' = a''_{ij} = \frac{a'_{ij} - a'_{iJ}}{||A||_i}, \tag{3.26}$$

onde a norma da linha i é

$$\|A\|_i = \sum_{j=1}^{m} (a'_{ij} - a'_{iJ})^2. \tag{3.27}$$

Feita a normalização, o algoritmo gera clusters *estáveis* separadamente para as linhas e para as colunas da matriz de dados. Esses clusters são gerados a partir de algum método de clusterização que possua a característica de determinar automaticamente o número de clusters e disponha de um critério de estabilidade. O método escolhido pelos autores foi uma clusterização hierárquica denominada *Superparamagnetic Clustering (SPC)* (Blatt et al., 1996). Uma vez gerados os clusters iniciais, o processo é iterado para cada sub-matriz formada pelo par de conjuntos de linhas e de colunas, até que novos clusters não possam ser formados.

Os autores relataram que, com esse procedimento, novas relações, nunca antes obtidas, foram encontradas em duas bases de dados de microagrupamento gênico, que poderiam sustentar novas interpretações para processos biológicos.

3.4.3 Algoritmo de Cheng & Church

A popularização do termo biclusterização e dos algoritmos de biclusterização com valores coerentes surgiu com o trabalho de Cheng e Church (Cheng & Church, 2000). Além de introduzirem a fórmula do RQM (Eq. 3.20), os autores também criaram o conceito de *δ-bicluster*. Um δ-bicluster é todo bicluster que possui RQM menor que um limiar pré-estabelecido (δ). E o problema proposto por Cheng e Church é o de encontrar um conjunto de δ-biclusters que possuam volume máximo, minimizando a sobreposição entre eles.

Nesse artigo, eles propuseram também uma heurística construtiva para encontrar P δ-biclusters em uma base de dados. Essa heurística, denominada *Cheng & Church*, ou *CC*, se baseia em três passos distintos. Os passos são aplicados sequencialmente, partindo de um bicluster inicial contendo todas as linhas e colunas da matriz de dados original.

O primeiro passo, denominado *Remoção de Múltiplos Nós*, com *nó* sendo o termo utilizado para definir linhas ou colunas de uma matriz de dados, consiste em encontrar o conjunto de linhas que tenham um RQM maior que $\alpha \times H(I, J)$, ou seja RQM do bicluster acrescido de uma taxa, com $\alpha > 1$, removendo simultaneamente todas as linhas que atendam essa condição. Note que, nesse ponto, é calculado o RQM apenas da linha utilizando o modelo de coerência do bicluster, e esse valor é então comparado ao RQM do bicluster completo.

Após esse passo, as médias das linhas, colunas e $H(I, J)$ são recalculados, e o processo é então repetido para a remoção de colunas. Esses dois itens são repetidos até que, ou $H(I, J)$ do bicluster se torne menor que δ ou não existam mais linhas ou colunas que atendam à condição de remoção. Esse passo está detalhado em Alg. 1. Em Cheng & Church (2000), o processo também era interrompido

caso o volume do bicluster atingisse um limiar.

Algoritmo 1 Remoção de Múltiplos Nós.

Calcula a_{Ij}, a_{iJ}, a_{IJ} e $H(I, J)$ (ver Eq. 3.20);
Remove todas as linhas $i \in I$ que satisfazem:

$$\frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2 > \alpha \times H(I, J)$$

Recalcule a_{Ij}, a_{iJ}, a_{IJ} and $H(I, J)$;
Remove todas as colunas $j \in J$ que satisfazem:

$$\frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2 > \alpha \times H(I, J)$$

O segundo passo, chamado de *Remoção de Nó Único*, remove o nó – linha ou coluna – do bicluster que mais contribui para o RQM. Esse processo é repetido até que o bicluster atinja $H(I, J) < \delta$. O processo é ilustrado em Alg. 2.

Algoritmo 2 Remoção de Nó Único.

Recalcule a_{Ij}, a_{iJ}, a_{IJ} e $H(I, J)$ do bicluster obtido pelo Alg. 1;
enquanto $H(I, J) > \delta$ **faça**
    Remove o nó (linha ou coluna) que mais contribui para o RQM;
    Recalcule a_{Ij}, a_{iJ}, a_{IJ} e $H(I, J)$;
fim enquanto

Finalmente, o último passo, chamado *Inserção de Múltiplos Nós*, tem o objetivo de procurar os nós removidos nos passos anteriores e que, dado o estado atual do bicluster, possam ser inseridos novamente, com ganho na minimização de $H(I, J)$. O processo inicia inserindo todas as colunas que não pertençam ao bicluster e que tenham um RQM menor ou igual ao do próprio bicluster. Segundo Cheng & Church (2000), a inserção de tais nós não aumenta o valor de $H(I, J)$. Em seguida, repete-se a mesma busca por linhas que não pertençam ao bicluster. Finalmente, as linhas que ainda não pertençam ao bicluster são utilizadas mas com os valores multiplicados por -1, tornando o algoritmo capaz de também encontrar coerências negativas. Após cada um desses itens, as médias e o RQM são recalculados e o processo é repetido até que nenhuma mudança possa ser feita.

Após esses três passos, o algoritmo retorna um único bicluster que respeita a restrição do valor do resíduo, enquanto maximiza o volume. Para encontrar os biclusters subsequentes, os autores substituem, na base de dados, os valores pertencentes ao bicluster por números aleatórios, introduzindo assim um ruído que direciona o algoritmo a buscar por diferentes biclusters. Esse procedimento é repetido até obter o número de biclusters desejados.

Esse algoritmo apresentou bons resultados (Cheng & Church, 2000), encontrando biclusters com uma média de volume alto e a média do RQM dentro do limite estipulado. Porém, a variância desses valores é alta, ou seja, apenas nas primeiras iterações o algoritmo foi capaz de encontrar biclusters

Algoritmo 3 Inserção de Múltiplos Nós.

Calcula a_{Ij}, a_{iJ}, a_{IJ} e $H(I, J)$;
Insere todas as colunas $j \notin J$ que satisfazem:

$$\frac{1}{|I|}\sum_{i\in I}(a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2 \leq H(I, J)$$

Recalcule a_{Ij}, a_{iJ}, a_{IJ} e $H(I, J)$;
Insere todas as linhas $i \notin I$ que satisfazem:

$$\frac{1}{|J|}\sum_{j\in J}(a_{ij} - a_{Ij} - a_{iJ} + a_{IJ})^2 \leq H(I, J)$$

Insere todas as linhas $i \notin I$ que satisfazem:

$$\frac{1}{|J|}\sum_{j\in J}(-a_{ij} + a_{Ij} + a_{iJ} - a_{IJ})^2 \leq H(I, J)$$

Repete até que nada seja inserido.

com um volume grande de dados. Nas iterações subsequentes, o tamanho dos biclusters diminui gradativamente.

3.4.4 Modelo Plaid

A maioria dos algoritmos de biclusterização para biclusters com valores coerentes leva em conta apenas o modelo aditivo ou o multiplicativo e, também, não explora a interação que pode existir entre os biclusters. Ou seja, esses algoritmos não são capazes de representar um elemento a_{ij} de acordo com a soma das contribuições de diferentes biclusters a que esse elemento pertença.

Pensando nisso, Lazzeroni & Owen (Lazzeroni & Owen, 2002) criaram o modelo *Plaid*, que pode ser traduzido livremente para *modelo quadriculado*, por conta da sobreposição dos vários biclusters para definir um elemento, formando um padrão quadriculado. Nesse modelo, o elemento a_{ij} é definido por:

$$a_{ij} = \sum_{p=1}^{P} \theta_{ijp} \cdot \rho_{ip} \cdot \kappa_{jp}, \tag{3.28}$$

onde P é o número de biclusters, ou partições, θ_{ijp} é a contribuição do elemento a_{ij} no bicluster p, ρ_{ip} e κ_{jp} são variáveis binárias que possuem valor 1 caso a linha i ou a coluna j, respectivamente, pertençam ao bicluster p, ou 0 caso contrário.

Similarmente à equação do RQM (Eq.3.20), a função-objetivo do modelo Plaid é:

$$\min f(\theta, \rho, \kappa) = \frac{1}{2} \cdot \sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - \theta_{ij0} - \sum_{p=1}^{P}\theta_{ijp} \cdot \rho_{ip} \cdot \kappa_{jp})^2, \tag{3.29}$$

com θ_{ij0} sendo um parâmetro de ajuste da variabilidade dos valores que não é influenciado por nenhum bicluster.

Com esse modelo, é possível generalizar, através da variável θ_{ijp}, qualquer outro modelo visto

anteriormente. Basta fazer com que $\theta_{ijp} = \mu$ para se obter um bicluster constante, $\theta_{ijp} = \mu + \alpha_i$ ou $\theta_{ijp} = \mu + \beta_j$ para se definir os biclusters constantes em linha ou em coluna, e $\theta_{ijp} = \mu + \alpha_i + \beta_j$, para se encontrar biclusters com valores coerentes.

Os autores propõem um método iterativo para encontrar os valores de θ, ρ e κ para cada elemento e cada bicluster. O algoritmo começa determinando os valores para o bicluster $p = 1$ e, após determinar os valores para esse bicluster, procede até chegar em $p = P$. O valor inicial do parâmetro de ajuste de cada bicluster é definido como $\theta_{ijp} = 1$, e a pertinência de cada linha e coluna no bicluster como $\rho_{ip} = 0,5 \pm r$ e $\kappa_{jp} = 0,5 \pm r$, com r sendo uma variável aleatória com valor baixo. Em seguida, ρ_{ip} e κ_{jp} são atualizados através das seguintes equações:

$$\rho_{ip} = \frac{\sum_j \theta_{ijp} \cdot \kappa_{jp} \cdot Z_{ij}}{\sum_j \theta_{ijp}^2 \cdot \kappa_{jp}^2}, \tag{3.30}$$

e

$$\kappa_{jp} = \frac{\sum_j \theta_{ijp} \cdot \rho_{ip} \cdot Z_{ij}}{\sum_j \theta_{ijp}^2 \cdot \rho_{ip}^2}, \tag{3.31}$$

onde Z_{ij} é o valor residual do bicluster anterior, ou a_{ij} no caso da primeira iteração.

Ao obter estabilidade, o algoritmo passa a atualizar os valores de θ, ρ, κ, nessa ordem, por um determinado número de iterações, utilizando os valores mais atuais de cada variável. A atualização de θ é dada pela minimização de

$$Q = \frac{1}{2} \sum_{i,j} (Z_{ij} - (\mu + \alpha_i + \beta_j)\rho_i\kappa_j)^2, \tag{3.32}$$

sujeito a

$$\sum_i \rho_i^2 \alpha_i = \sum_j \kappa_j^2 \beta_j = 0, \tag{3.33}$$

e com

$$\mu = \frac{\sum_{i,j} \rho_i \kappa_j Z_{ij}}{(\sum_i \rho_i)^2 (\sum_j \kappa_j)^2}, \tag{3.34}$$

$$\alpha_i = \frac{\sum_j (Z_{ij} - \mu\rho_i\kappa_j)\kappa_j}{\rho_i \sum_j \kappa_j^2}, \tag{3.35}$$

$$\beta_j = \frac{\sum_j (Z_{ij} - \mu\rho_i\kappa_j)\rho_i}{\kappa_j \sum_i \rho_i^2}. \tag{3.36}$$

Os valores de ρ e κ são atualizados pelas Eqs. 3.30 e 3.31. Os autores encontraram resultados

interpretáveis em bases de dados de expressão gênica, de dados nutricionais e operações de câmbio.

3.4.5 xMotif

Para os biclusters de evolução coerente, o algoritmo mais popular atualmente é o *xMotif* (Murali & Kasif, 2002). Nesse artigo, os autores definem *xMotif* como sendo um subconjunto de linhas e colunas da matriz original de dados que satisfaz as seguintes condições:

- O tamanho do subconjunto de colunas é pelo menos uma fração α da quantidade total de colunas na matriz original.
- Em todas as linhas do subconjunto, cada elemento é conservado por todas as colunas do subconjunto, isto é, os valores de cada linha apresentam o mesmo estado.
- Para todas as linhas que não estão presentes no *xMotif*, o estado é conservado em no máximo uma fração β do número de colunas do *xMotif*.

O objetivo desse algoritmo é encontrar, então, o *xMotif* que apresenta o maior número de linhas dentro do conjunto de dados. Para encontrar o maior *xMotif*, os autores criaram um algoritmo probabilístico que consistia em sortear aleatoriamente uma coluna inicial para fazer parte do *xMotif* e, em seguida, com um processo iterativo, determinar as linhas e colunas que satisfazem as condições necessárias para a criação de um *xMotif*. Ao final de várias tentativas, o maior *xMotif* encontrado é retornado.

3.5 Meta-heurísticas para biclusterização

Apesar de a maioria dos métodos apontados na seção anterior apresentar resultados satisfatórios, eles não têm garantia de resultados ótimos, deixando em aberto a possibilidade de obter melhores resultados. Tendo em vista que a maioria das abordagens parte de heurísticas construtivas, as decisões locais a cada passo não podem garantir a obtenção do ótimo global ao final. Uma forma de obter uma melhor capacidade de exploração do espaço de busca, utilizando informações globais no processo de decisão, é através do uso de meta-heurísticas (Glover & Kochenberger, 2003). As meta-heurísticas são heurísticas combinadas com o objetivo de explorar o espaço de busca de tal forma a obter um desempenho melhor do que uma heurística isolada. Conforme relatado anteriormente, o foco desta tese será o de encontrar biclusters com valores coerentes aditivos, no mesmo modelo proposto por Cheng & Church. Por esse motivo, essa seção fará uma revisão bibliográfica de algumas abordagens meta-heurísticas que têm o objetivo de encontrar esse tipo de bicluster.

3.5.1 Algoritmo multi-objetivo de Mitra & Banka

Analisando as heurísticas apresentadas na seção anterior, é possível verificar que a maioria delas, e principalmente o algoritmo CC, possui dois objetivos claros: encontrar um bicluster que mais se aproxime do modelo ideal, ou seja, minimizar o erro quadrático, e que esse bicluster apresente máximo volume. Esses dois objetivos tendem a ser conflitantes, tendo em vista que, quanto maior o bicluster, maiores são as chances desse ter ruído que aumenta o erro quadrático.

Partindo desse conflito, Mitra & Banka (2006) reformularam o problema de encontrar biclusters com valores coerentes como um problema multi-objetivo. Dessa forma, eles buscam um conjunto de biclusters *não-dominados* dentro da base de dados. Um conjunto de soluções não-dominadas é a definição dada quando não é possível determinar qual a melhor solução dentro do conjunto de soluções, pois para cada solução existe pelo menos uma função-objetivo melhor e uma pior, quando comparada com qualquer outra solução desse conjunto (Deb et al., 2002).

O método utilizado para esse fim foi uma adaptação do algoritmo conhecido como *NSGA-II* (Deb et al., 2002), do inglês *Non-Dominated Sorting Genetic Algorithm*, que se baseia em um *algoritmo genético* (Goldberg, 1989). Um algoritmo genético é uma técnica de busca inspirada em processos evolutivos utilizando conceitos de população, reprodução e mutação. A metodologia principal do algoritmo NSGA-II é ilustrada em Alg. 4 e as adaptações feitas por Mitra & Banka (2006) serão explicadas a seguir.

Algoritmo 4 Algoritmo NSGA-II utilizado para biclusterização por Mitra & Banka (2006).

1) Gera população inicial de tamanho P.
2) Aplica busca local para melhorar as soluções da população inicial.
3) Calcula o valor de cada função-objetivo para cada solução.
4) Gera o *ranking* das soluções utilizando um critério de dominância.
5) Calcula a *distância de agrupamento* das soluções que são não-dominadas entre si.
6) Seleciona soluções para participarem da reprodução.
7) Aplica reprodução e mutação nas soluções selecionadas, gerando P *soluções-filhas*.
8) Junta a população *progenitora* com a população *filha* em uma única população de tamanho $2P$.
9) Gera o rankeamento e calcula a distância de agrupamento nessa população.
10) Seleciona P soluções para participarem da próxima geração, retornando ao passo 2 até atingir o critério de parada.

Para resolver um problema utilizando algoritmos genéticos, um dos requisitos iniciais é definir como uma solução desse problema deverá ser representada. Em Mitra & Banka (2006), um bicluster foi representado por um vetor binário de tamanho $n + m$, onde o valor 1 indica que a linha ou coluna correspondente àquela posição pertence ao bicluster, e 0, caso contrário. A população de soluções iniciais contém P indivíduos e é gerada distribuindo aleatoriamente, de forma uniforme, valores 0 e 1 para cada um dos P indivíduos, sendo que cada indivíduo é um vetor representando um bicluster.

A busca local utilizada para melhorar as soluções obtidas são a *Remoção de Nó Único* (Alg. 2) e *Inserção de Múltiplos Nós* (Alg. 3) do algoritmo CC, aplicados sequencialmente.

Para avaliar os biclusters gerados, os autores propuseram duas funções-objetivo distintas:

$$\max f_1 = \frac{|I| \times |J|}{n \times m}, \tag{3.37}$$

$$\max f_2 = \begin{cases} \frac{H(I,J)}{\delta} & \text{se } H(I,J) \leq \delta \\ 0 & \text{c.c.,} \end{cases} \tag{3.38}$$

A primeira função-objetivo, Eq. 3.37, corresponde ao volume do bicluster, normalizado pelo tamanho total da base de dados. O segundo objetivo, Eq. 3.38, diz respeito ao RQM, e maximiza $H(I, J)$, normalizado pelo limiar δ, enquanto esse resíduo não ultrapassar o limiar. Caso ultrapasse, essa equação atribui o valor 0 para essa função. Dessa forma, essa segunda função-objetivo também é de maximização, somente limitada pelo limiar δ. Embora esse objetivo pareça destoar do que foi descrito anteriormente quanto a busca por um bicluster com RQM mínimo, os autores desse método optaram por maximizar o volume através da maximização do RQM, enquanto este possuir um valor menor do que δ, tornando a solução factível.

Com as soluções avaliadas, a *distância de agrupamento* (Deb et al., 2002) entre as soluções obtidas é calculada. Essa medida de distância tem o objetivo de, em uma população de soluções, medir a densidade das soluções no espaço de objetivos, usando o vetor de valores de funções-objetivo para calcular tal densidade. Com essa medida, é possível selecionar um conjunto dentro da população de soluções de forma a maximizar a diversidade. Para esse algoritmo, a seleção utilizando essa medida de distância começa dividindo as soluções em *ranks*, definidos pelo grau de dominância.

Esse ranking é feito da seguinte forma: todos os indivíduos não-dominados na população passam a pertencer ao *rank* 1. Excluindo essas soluções-candidatas, todas as soluções-candidatas restantes e que passam a ser não-dominadas farão parte do *rank* 2, e assim por diante, até que todos os indivíduos recebam sua posição no ranking. Feito esse procedimento, a seleção de indivíduos para realizar o procedimento de reprodução é feita a partir de um *torneio binário*, em que dois indivíduos são selecionados aleatoriamente. O indivíduo que tiver o menor *rank* será o escolhido para reprodução. Caso os dois possuam o mesmo *rank*, aquele que tiver o maior valor de distância de agrupamento será escolhido e, em caso de novo empate, a decisão é feita aleatoriamente.

Após o processo de seleção, tomam-se os P indivíduos dois a dois para o processo de reprodução. Esse processo consiste em, dadas duas soluções-candidatas p_1 e p_2, selecionar um mesmo ponto aleatório de corte nas duas soluções-candidatas que as divida em duas partes. Em seguida, faz-se uma recombinação gerando duas soluções-*filhas*, sendo que a primeira solução-filha é formada pela

primeira parte da solução p_1 e a segunda parte da solução p_2. A segunda solução-filha é o oposto. Esse processo é ilustrado na Fig. 3.7.

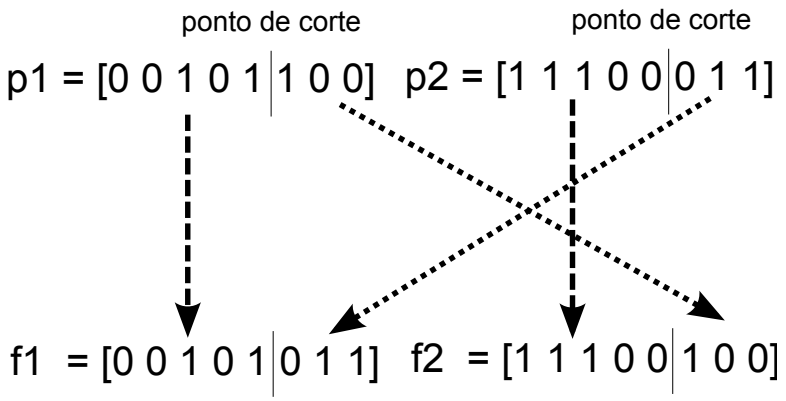


Fig. 3.7: Exemplo de recombinação entre duas soluções ($p1$ e $p2$) que geram dois filhos ($f1$ e $f2$) a partir do ponto de corte, definido aleatoriamente.

A ideia desse procedimento é que, diante da seleção de bons indivíduos, o *material genético* desses seja combinado obtendo-se soluções melhores. Após o processo de reprodução, é aplicada a mutação nas novas soluções geradas. Esse processo faz a troca de um determinado elemento de cada vetor-solução (de 0 para 1, ou vice-versa) com uma probabilidade baixa, definida pelo usuário.

Finalmente, as populações *pai* e *filha* são combinadas em uma única população e cada vetor-solução é *rankeado*. Após a definição do *rankeamento*, os elementos do primeiro *rank* são selecionados para participarem da próxima geração, até um limite de P soluções. Caso o primeiro *rank* tenha mais indivíduos que P, é utilizado o critério de distância de agrupamento, eliminando aqueles com menor distância. Se o *rank* inicial possuir menos que P soluções, os próximos *ranks* são utilizados para preencher a quantidade restante, até que se atinja o tamanho definido.

A partir desse ponto, o algoritmo repete todos os procedimentos até um determinado número de iterações. Esse algoritmo se baseia no fato de que, dada a pressão seletiva, partes de soluções-candidatas boas permanecerão ativas nas gerações futuras, enquanto partes de soluções-candidatas consideradas ruins serão descartadas. Tendo apenas as partes boas das soluções-candidatas, o espaço de busca se torna reduzido e o algoritmo passa por um processo probabilístico tentando encontrar combinações dessas partes que levem a soluções-candidatas ainda melhores.

Os resultados obtidos por essa abordagem apresentam uma melhora em relação ao tamanho dos biclusters obtidos pelo algoritmo CC, sob as mesmas condições, e um RQM médio também maior, embora dentro do limite estipulado.

3.5.2 Algoritmo BIC-aiNet

Uma outra forma de tratar um problema multi-objetivo, quando se sabe a importância de cada função-objetivo, é transformá-lo em um problema mono-objetivo composto pela soma ponderada de cada uma das funções-objetivo a serem otimizadas. Partindo desse princípio, de Castro et al. (2007b,a,c); de Castro et al. (2010) desenvolveram o algoritmo denominado *BIC-aiNet* – do inglês, *Artificial Immune Network for Biclustering* – cuja função-objetivo é composta de um termo referente ao resíduo, um termo ponderando o número de linhas e outro ponderando o número de colunas.

Essa meta-heurística foi desenvolvida de acordo com a já conhecida *aiNet* (de Castro & Von Zuben, 2000, 2001; de França et al., 2010) – *Artificial Immune Network* – que tem sua inspiração no funcionamento dos sistemas imunológicos. Em resumo, o sistema imunológico, para combater corpos estranhos ao organismo (antígenos), possui um sistema eficiente de aprendizado para detectar e destruir tais antígenos, criando anticorpos especializados para isso. Uma característica interessante desse sistema é a multi-modalidade, ou seja, a capacidade de encontrar e manter múltiplas soluções ótimas locais. Essa capacidade permite que o organismo mantenha o conhecimento sobre os diversos antígenos que já reconheceu, para não ser atacado por estes novamente. Para o caso específico de biclusterização, essa característica promove o encontro de diversos biclusters com características, e qualidades, diferentes.

Da mesma forma que o algoritmo de Mitra & Banka (2006) buscava pelo conjunto de biclusters paralelamente, evoluindo uma população de biclusters, o algoritmo BIC-aiNet também evolui vários biclusters ao mesmo tempo, com a diferença de que cada bicluster evolui de forma independente dos outros, criando o conceito de multi-população, ou seja, cada bicluster se torna representante de sua própria *micro*-população, e define seu próprio processo evolutivo em um raio de atuação. Esse raio de atuação é definido pela mutação utilizada. Apesar de evoluírem de forma independente, os biclusters se conectam através de uma rede que tem a função de diversificar a região que cada bicluster cobre da base de dados original.

O primeiro passo para o entendimento do algoritmo é a forma como ele representa um bicluster dentro da população. Diferente do algoritmo de Mitra & Banka, os biclusters da BIC-aiNet são representados por dois vetores de tamanhos variáveis, com valores inteiros que funcionam como um mapa entre o bicluster e a base de dados. Cada valor desses vetores representa o índice de uma linha ou coluna da base original.

Inicialmente, a população é formada por k biclusters gerados aleatoriamente e cada um contendo apenas 1 linha e 1 coluna. Ao ser iniciado, o algoritmo entra no laço principal, onde são gerados n_c *clones* – cópias idênticas – para cada bicluster. Cada um desses clones passa por um processo de mutação que possibilita três movimentos diferentes, escolhidos aleatoriamente e com igual probabilidade:

- **Inserir uma linha:** insere aleatoriamente uma linha que ainda não pertença ao bicluster.
- **Inserir uma coluna:** insere aleatoriamente uma coluna que ainda não pertença ao bicluster
- **Remover uma linha ou uma coluna:** remove aleatoriamente uma linha ou uma coluna do bicluster.

É fácil perceber que esse processo de mutação favorece a inserção de elementos (linhas ou colunas) no bicluster, uma vez que essa operação tem $2/3$ de chance de ocorrer, enquanto a remoção de linhas e colunas, que é encarada como um processo único, tem apenas $1/3$ de chance. Esse favorecimento ocorre para ajudar na maximização do volume, um dos objetivos.

Após o processo de clonagem e mutação, cada bicluster novo é avaliado e comparado com o bicluster-*pai* que o gerou. Caso esse seja melhor, substitui o bicluster-pai. Conforme mencionado anteriormente, para fazer a avaliação de um bicluster foi utilizada uma função-objetivo unificada, ponderando o objetivo de minimizar o RQM e maximizar o volume. A função-objetivo é representada pela minimização da seguinte função, adaptada de Divina et al. (2006):

$$F(A(I,J)) = \frac{H(I,J)}{\delta} + \frac{w_r \times \delta}{|I|} + \frac{w_c \times \delta}{|J|}, \tag{3.39}$$

lembrando que $H(I,J)$ é o RQM do bicluster definido pelos conjuntos de linhas I e colunas J (ver Eq. 3.20), δ é o peso atribuído à minimização do resíduo, w_r e w_c são, respectivamente, os pesos atribuídos à maximização da quantidade de linhas e colunas do bicluster e $|\zeta|$ representa a quantidade de elementos no conjunto ζ.

Conforme é possível perceber, ao minimizar a função dada na Eq. 3.39, o RQM é minimizado, amortecido por um peso, representado pelo máximo RQM desejado – δ, e o volume é maximizado proporcional também a δ e a um peso específico para o número de linhas e o número de colunas, uma vez que o aumento do volume está associado ao aumento do RQM.

Após o processo de seleção baseado na Eq. 3.39, o processo de clonagem e mutação é repetido até uma certa condição de parada, geralmente associada ao número de iterações. Em determinados momentos, também é feito um processo nativo de algoritmos imuno-inspirados: a manutenção da diversidade da população. Essa manutenção é feita através de dois processos: supressão de biclusters e inserção de novos biclusters.

A supressão tem o objetivo de verificar biclusters dentro da população que estejam muito próximos uns dos outros, e consequentemente caminhando para o mesmo ótimo local, e eliminar aqueles que menos se aproximam do ótimo. Com isso, é possível remover as redundâncias do processo de busca de tal forma que se reduza o custo computacional por iteração do algoritmo. Esse processo é feito comparando cada possível par de biclusters da população entre si e, caso a semelhança entre eles

seja maior que um determinado limiar, aquele que tiver uma avaliação da função-objetivo pior será removido da população.

Para medir a semelhança entre dois biclusters, é medido o volume do bicluster obtido pela interseção desses dois biclusters, ou seja, o volume da *sobreposição* dos biclusters (Eq. 3.1).

A inserção de novos biclusters na população tem o objetivo de explorar regiões do espaço de busca que não estão sendo exploradas, no momento, pela população. Com esse procedimento, é possível aumentar a *cobertura* que o conjunto de biclusters possui em relação à base de dados original. Esse procedimento funciona gerando n_{ins} novos biclusters na população, gerados da mesma forma que a população inicial, mas dando preferência aos elementos que ainda não pertençam a nenhum bicluster.

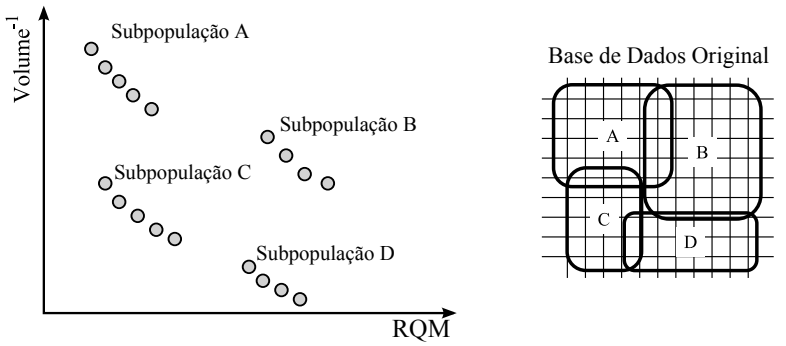
Após esses passos, o tamanho da população pode sofrer alterações, o que é uma característica a ser destacada dessa classe de algoritmos. Com esses procedimentos, o tamanho da população tende a se auto-ajustar, buscando um tamanho apropriado ao número de biclusters que consegue cobrir corretamente a base.

Esse algoritmo foi utilizado em algumas aplicações pouco usuais em biclusterização, como mineração de textos (de Castro et al., 2007b; de Castro et al., 2010) e filtragem colaborativa (de Castro et al., 2007a,c), obtendo resultados expressivos e significativos nas duas aplicações.

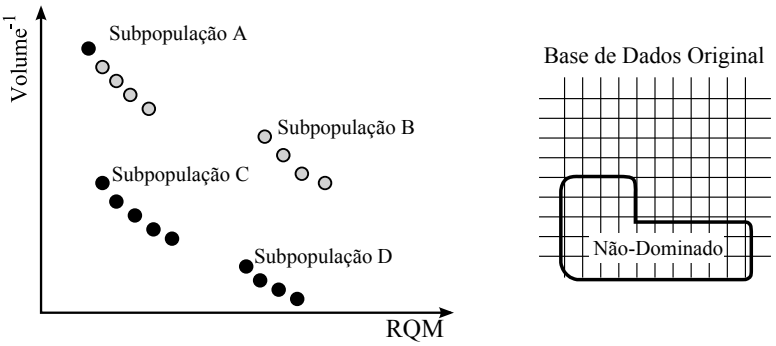
3.5.3 Algoritmo MOM-aiNet

Apesar de sua natureza claramente multi-objetivo, Coelho et al. (2008, 2009b) perceberam que o problema de biclusterização pode ter uma interpretação mais complexa. Segundo a abordagem multi-objetivo de Mitra & Banka (2006), o algoritmo procurava por uma única fronteira de soluções não-dominadas e, nessa fronteira estava contida o conjunto de biclusters, em que cada um deles poderia representar qualquer região da base de dados. Nesse caso, o problema multi-objetivo era o problema de encontrar múltiplos biclusters. Porém, a multi-objetividade pode ser interpretada, de forma alternativa, como a busca por diferentes versões de um mesmo bicluster de referência, ou seja, uma fronteira de soluções não-dominadas representa versões diferentes de um mesmo bicluster com pesos diferentes para cada uma das funções-objetivo.

Consequentemente, em uma população de biclusters, como aquela apresentada na Fig. 3.8(a), a interpretação de Mitra & Banka da multi-objetividade levaria à filtragem de biclusters de tal forma que a população seguinte convergiria para a situação ilustrada na Fig. 3.8(b) (os círculos pretos são selecionados por corresponderem a soluções não-dominadas). Como é possível verificar, a nova população tem uma cobertura da base de dados muito menor do que o conjunto original, pois biclusters de diferentes áreas foram eliminados. Através dessas ilustrações, é possível perceber que o problema multi-objetivo de biclusterização pode ser tratado em separado para cada bicluster, ou seja, o algoritmo pode optar por lidar com o aspecto multimodal desse problema.



(a) Representação pictórica da região da matriz de dados coberta por cada bicluster obtido. Na prática, as regiões cobertas podem não ser contíguas, como no exemplo.



(b) Representação pictórica da região da matriz de dados coberta pelo conjunto não-dominado de biclusters, obtidos através da figura anterior. Na prática, as regiões cobertas podem não ser contíguas, como no exemplo.

Fig. 3.8: Representação pictórica da cobertura da matriz de dados original pelo conjunto de biclusters gerado (a) e pelo conjunto de biclusters não-dominados (b).

Para tanto, Coelho et al. (2008, 2009b) criaram o algoritmo chamado MOM-aiNet – do inglês, *Multi-Objective Multipopulation Artificial Immune Network for Biclustering* – que, da mesma forma que a BIC-aiNet, é baseado na aiNet. A escolha por essa meta-heurística foi feita principalmente por causa de sua característica multimodal, definida através do conceito de multipopulação.

Esse algoritmo segue o mesmo funcionamento básico da BIC-aiNet, tendo a mesma representação do bicluster através de dois vetores de tamanho variável. Porém, após o processo de clonagem e mutação, não mais será escolhido apenas o melhor clone. Serão escolhidos todos os clones que obedecem o critério de não-dominância, ao minimizar resíduo e maximizar volume, gerando então diversas *micro-populações*. Nas iterações subsequentes, o processo de clonagem escolhe n_{clones} biclusters dentro de cada micro-população para serem clonados e participarem da mutação.

A mutação ocorre da mesma forma que na BIC-aiNet, sendo escolhida uma entre três ações com igual probabilidade: inserir uma linha, inserir uma coluna ou remover uma linha ou coluna.

Após a mutação, o conjunto de biclusters não-dominados é mantido dentro dessa micro-população se o tamanho dela for menor ou igual a n_{clones}. Caso contrário, a seleção por *distância de agrupamento* é efetuada, da mesma forma que no algoritmo de Mitra & Banka, para obter uma população total de n_{clones} biclusters.

Essencialmente, esse algoritmo funciona da mesma forma que a BIC-aiNet, estando as diferenças na avaliação e seleção da população de clones. Essa seleção é feita utilizando critérios multi-objetivos de dominância, mantendo toda a população de clones não-dominados. Essa diferença causa uma inflação no tamanho total da população, uma vez que várias micro-populações são mantidas por cada região explorada. O processo de clonagem seleciona n_{clones} biclusters de cada micro-população para copiá-los e sofrer um processo de mutação igual ao da BIC-aiNet: inserir uma linha, inserir uma coluna ou remover uma linha ou coluna.

Além dos objetivos principais, Coelho et al. (2008, 2009b) definiram duas restrições para limitar o crescimento da micro-população e respeitar os critérios de δ-bicluster. O primeiro é justamente a restrição de valor do RQM, definido por δ, em que o bicluster que ultrapassar tal limiar é imediatamente eliminado. A outra restrição tem relação com biclusterização de matrizes esparsas: ela define a mínima taxa de valores não-faltantes que uma linha ou coluna de um bicluster pode ter. Esse critério é aplicado antes de efetuar a mutação, pré-selecionando apenas as linhas e colunas que respeitam esse critério ao serem inseridas ou removidas.

A supressão e inserção de novos biclusters são feitas de maneira similar ao procedimento adotado para a BIC-aiNet, sendo diferente apenas no método de supressão, em que apenas os indíviduos de maior volume de cada micro-população – os mais representativos – são comparados e, ao encontrar dois biclusters similares, as duas micro-populações às quais eles pertencem são mescladas e reduzidas ao tamanho de n_{clones} bicluster, seguindo o mesmo critério da seleção de clones.

Em Coelho et al. (2009a), os autores propuseram melhorias para esse algoritmo, introduzindo uma função de *reparação* dos clones infactíveis, através do procedimento de Remoção de Múltiplos Nós do algoritmo CC, e uma busca local para melhorar os biclusters mais representativos de cada micro-população, utilizando o procedimento de Inserção de Múltiplos Nós do CC. Esse algoritmo foi denominado MOM-aiNet+ para diferenciar do seu predecessor.

Esse algoritmo obteve resultados bons, quando comparado com outras abordagens, em relação à cobertura total da base de dados, enquanto se manteve competitivo em relação à média dos RQMs e volume dos biclusters.

3.5.4 Algoritmo bicACO

Recordando o algoritmo proposto por Cheng & Church, o método iniciava a busca por um bicluster supondo que todas as linhas e colunas da base de dados pertenciam a ele. Em seguida, eliminava

as linhas e colunas com um alto valor residual até obter um bicluster com o RQM abaixo do limiar estipulado. A escolha de quais linhas e colunas seriam removidas é feita de forma *gulosa*, ou seja, aquela que apresenta o valor residual mais alto é removida primeiro. Embora essa sequência traga, em média, bons resultados, ela não garante a otimalidade, isto é, pode existir uma sequência diferente capaz de encontrar um melhor resultado.

A busca pela sequência ótima é um problema combinatório e intratável computacionalmente de forma exata. Uma forma de tentar encontrar sequências ótimas é amostrando um conjunto de sequências geradas aleatoriamente e, utilizando essa informação, gerar uma função de densidade de probabilidade para definir a probabilidade de uma certa sub-sequência pertencer à solução ótima. Uma meta-heurística que trata problemas combinatórios dessa forma é o *Ant Colony Optimization* (ACO) (Dorigo, 1992).

O ACO é um algoritmo inspirado no comportamento das colônias de formigas em busca de alimento e foi proposto, inicialmente, por Dorigo (1992) para resolver o problema do Caixeiro Viajante. Os detalhes dessa meta-heurística serão apresentados no próximo capítulo, quando será introduzido um novo algoritmo para biclusterização baseado nesse conceito. Em resumo, o ACO inspira-se na tentativa de uma colônia de formigas encontrar o caminho mais rápido entre o seu ninho e alguma fonte de alimento, de forma a minimizar o tempo gasto com a coleta de alimentos. Para isso, as formigas começam a busca caminhando aleatoriamente em torno de seu ninho, até encontrar alguma fonte de alimento. Ao encontrar, uma formiga percorre o caminho de volta depositando uma substância denominada *feromônio*, que serve para comunicar às outras formigas sobre o caminho encontrado. As outras formigas tendem a seguir essa trilha com uma probabilidade proporcional à intensidade do feromônio. Como as menores trilhas são percorridas mais rapidamente, o depósito de feromônio é intensificado nelas, fazendo com que quase todas as formigas caminhem pelo melhor caminho encontrado no processo.

Discretizando o espaço percorrido pelas formigas, a matriz de feromônio pode ser interpretada como dados estatísticos da qualidade de pequenos *pedaços* do caminho. Esses dados, por sua vez, são interpretados como a probabilidade daquele pedaço pertencer ao melhor caminho possível.

Essa meta-heurística constrói uma solução iterativamente, em que a decisão do próximo passo a ser tomado é definida pela matriz de feromônio. Ou seja, quanto maior o valor na matriz de feromônio, maior a probabilidade de tal passo ser escolhido.

O primeiro algoritmo de biclusterização a adaptar a meta-heurística ACO foi feito em de França et al. (2008) e denominado bicACO – do inglês, *Biclustering Ant Colony Optimization*. Nesse algoritmo, *k formigas* – denominação de cada *agente construtor* de uma solução – iniciam a busca pela solução com a matriz de dados completa. Em cada passo, cada uma delas remove uma linha ou uma coluna, probabilisticamente, de seu respectivo bicluster. Esses passos se repetem até que o resíduo de

cada bicluster atinja um valor δ ou o volume atinja um valor mínimo.

Cada formiga terá sua própria tabela de feromônio, diferente do algoritmo ACO original, definindo a probabilidade de um elemento do bicluster ser removido. Dessa forma, para escolher qual elemento remover no passo atual, a seguinte equação é utilizada:

$$p_i = \frac{\tau_i^\alpha \eta_i^\beta}{\sum_{j=1}^{n+m} \tau_j^\alpha \eta_j^\beta}, \tag{3.40}$$

onde τ_i é o nível de feromônio do elemento (linha ou coluna) i, η_i é a *qualidade* do elemento i, α e β são pesos ponderando a importância de τ e η, n e m indicam a dimensão atual do bicluster.

O termo η, nesse caso, é proporcional ao ganho que se obtém ao remover tal elemento do bicluster, e é calculado por:

$$\eta_i = \begin{cases} \frac{1}{|J|} \sum_{j \in J} r_{ij}^2 & \text{se } i \text{ for índice da linha} \\ \frac{1}{|I|} \sum_{j \in I} r_{ji}^2 & \text{se } i \text{ for índice da coluna} \end{cases}, \tag{3.41}$$

lembrando que r_{ij} é o resíduo da linha i e coluna j.

Esse termo ajuda principalmente nos passos iniciais, em que não existe informação alguma, acelerando o processo de geração da tabela de feromônio. A tabela de feromônio inicia com um valor igual e pré-fixado para todos os elementos e, ao final da construção de um bicluster, é alterada da seguinte forma:

$$\tau_i = \tau_i + \rho.(\Delta\tau - \tau_i) \quad \text{para todo elemento} \quad i \in B(I, J), \tag{3.42}$$

$$\Delta\tau = \frac{\frac{|I||J|}{H(I,J)}}{\sum_{j=1}^{k} \frac{|I_j||J_j|}{H(I_j,J_j)}}, \tag{3.43}$$

onde $\Delta\tau$ é o valor a ser incrementado ao feromônio, ρ é uma taxa de decaimento para impedir a convergência, I_j e J_j são referentes aos conjuntos de linhas e colunas do bicluster gerado pela formiga j.

Após a atualização das taxas de feromônio, o processo se repete do começo até um certo número de iterações. Ao final, todos os k biclusters são retornados.

Os resultados obtidos pelo bicACO foram melhores que o algoritmo de Cheng & Church em duas bases de dados de expressão gênica, embora necessite de um tempo computacional elevado.

3.6 Síntese do capítulo

Este capítulo apresentou os principais conceitos do problema de biclusterização, assim como algumas aplicações práticas desse problema. Inicialmente, foram levantandos um histórico dessa área de estudo e o porquê da necessidade de tal abordagem para suprir as limitações do problema de clusterização.

Em seguida, as diferentes formulações para esse problema foram explicadas, junto com as notações básicas e medidas de qualidade que serão utilizadas na sequência desta tese. O restante do capítulo concentrou-se em resumir o conceito das heurísticas para os diferentes tipos de biclusterização e, também, meta-heurísticas para os casos da chamada δ-biclusterização, que será objeto de estudo do próximo capítulo.

Foi possível perceber que os algoritmos apresentados, principalmente os de δ-biclusterização, buscam um conjunto de biclusters que apresentem relações importantes da base de dados, as quais não são possíveis de obter com clusterização. Muitos dos algoritmos focam, principalmente, na forma de tratar as possíveis funções-objetivo e como mensurar, de forma equilibrada, a qualidade do conjunto de biclusters, visando obter volume máximo enquanto o RQM é restrito a um determinado limiar.

Porém, conforme será mostrado no próximo capítulo, esses algoritmos nem sempre conseguem obter um conjunto de biclusters diversificados, no sentido de serem capazes de maximizar a cobertura da base de dados. Esse problema será solucionado com a criação de um novo algoritmo capaz de suprir essa demanda, mantendo as outras características (volume e RQM) já presentes nos demais algoritmos.

Capítulo 4

Biclusterização baseada em inteligência de enxame

O capítulo anterior ilustrou os principais aspectos do problema de biclusterização, bem como indicou que existem diversas formas de avaliar a qualidade dos biclusters encontrados. Essas formas de avaliação são baseadas em diversas características distintas que um bicluster ou um conjunto de biclusters pode ter. Dependendo da aplicação, a otimização de mais do que um objetivo, ou um compromisso entre vários desses objetivos, pode ser a melhor opção.

Além disso, os objetivos dessa otimização geralmente são independentes para cada um dos biclusters, no conjunto encontrado, ou seja, cada região da base de dados retorna um bicluster com um compromisso diferente entre objetivos, impossibilitando que se crie um parâmetro global que defina um peso ou limiar para estabelecer o compromisso desejado.

Dessa forma, o que se deseja é um algoritmo de biclusterização com a capacidade de realizar buscas paralelas e independentes, e que procure por biclusters em diferentes regiões da base de dados (visão local), podendo ou não haver sobreposições entre elas (visão global), de forma a encontrar o conjunto de biclusters com melhor qualidade possível, definida pelos objetivos da aplicação, e que represente a maior parte possível da base de dados, sem comprometer os objetivos.

Uma metodologia que prevê esse tipo de comportamento, em que soluções são exploradas, em paralelo, com compartilhamento de informações globais, é conhecida por *Inteligência de Enxame* ou *Inteligência Coletiva*. Essa metodologia define um conjunto de técnicas baseadas em *agentes* capazes de encontrar uma única solução de forma eficiente, independente e com boa qualidade. Apesar de realizarem a busca de forma independente, esses agentes contam com algum nível de comunicação direta ou indireta, dependendo do algoritmo, de forma que eles tenham informações globais para melhorar suas próprias soluções.

Como esses agentes devem ser simples e eficientes, em alguns problemas eles podem ser responsá-

veis por encontrar apenas uma solução parcial, que é então combinada com outras partes encontradas por outros agentes de forma a encontrar uma solução unificada e de boa qualidade. O importante para aplicar essa metodologia é que o agente seja capaz de encontrar novas informações do espaço de busca, através da *exploração*, e utilizar informações já abstraídas do espaço por ele ou por outros agentes, num processo denominado *explotação*.

Um dos algoritmos pertencentes a essa classe é conhecido como *Otimização baseada em Colônia de Formigas*, do inglês *Ant Colony Optimization* – ACO (Dorigo, 1992), já explicado brevemente na seção 3.5.4 do capítulo anterior. Esse algoritmo é inspirado no comportamento da busca que uma colônia de formigas faz por fontes de alimentos dentro de uma área em torno de seu ninho. Para tanto, esse problema foi modelado de forma similar ao conhecido Problema do Caixeiro Viajante (Kruskal Jr, 1956; Applegate, 2006), em que os possíveis caminhos são discretizados em forma de grafo e o objetivo é encontrar o menor caminho, de ida e volta, entre dois pontos. As formigas, conforme fazem uma busca por um caminho ótimo, depositam uma substância própria de forma a indicar que passou por determinado caminho. Dependendo da extensão do percurso, essa formiga poderá passar rapidamente por ele, e a cada passagem reforçar a quantidade dessa substância. Se o percurso for longo, a substância sofrerá mais com o efeito de evaporação, pois aumenta o intervalo entre duas passagens de formigas.

Com cada formiga (agente) percorrendo seu caminho de forma independente, logo o espaço discreto terá informações estatísticas da amostra de soluções geradas paralelamente, através da intensidade da substância depositada. Essa informação estatística pode ser utilizada pelas formigas, em um próximo momento, para direcionar a busca de cada uma delas em torno de regiões sub-ótimas. O objetivo é encontrar soluções ainda melhores. As formigas podem perceber essa informação parcialmente, ou seja, a cada passo da construção de uma solução, ou integralmente, antes de definir a sub-região de busca.

De modo geral, algoritmos desse tipo podem contribuir com a busca por biclusters em uma base de dados, caso sejam bem modelados e implementados. Este capítulo trata justamente da definição de uma abordagem baseada em ACO para a obtenção de δ-biclusters com volumes máximos e máxima cobertura dos dados. Primeiramente, os conceitos de Inteligência de Enxame e Otimização por Colônia de Formigas serão detalhados, porém, o foco principal será um novo algoritmo para biclusterização que será explicado em detalhe juntamente com experimentos em bases de dados reais.

4.1 Inteligência de enxame

O comportamento social de muitas ordens de insetos é de grande interesse em estudos biológicos e ecológicos, por suas notáveis características. Esses insetos são capazes de realizar tarefas complexas

distribuindo as tarefas entre indivíduos de uma colônia, de forma organizada e eficiente. O detalhe que desperta grande atenção é que essa distribuição é feita de forma descentralizada, ou seja, sem intermédio de um *líder*. Esse mecanismo é denominado auto-organização.

Juntamente com essa descentralização, existe também o fato de que cada indivíduo é limitado em seu arsenal de comportamentos possíveis e também em sua capacidade de aprendizado, tornando ainda mais essencial o trabalho em grupo. Exemplos de processos auto-organizados e descentralizados podem ser citados, como a construção de um ninho por uma colônia de cupins. Um cupim, individualmente, tem uma capacidade de ação limitada, além de sentidos limitados.

Apesar dessas limitações, que permitem classificar os cupins como trabalhadores cegos, o trabalho coletivo e auto-organizado torna possível o surgimento de estruturas complexas no ninho, de tal forma a apresentar pouca variação em suas propriedades (fluxo de ar, temperatura interna,...) mesmo quando a construção é expandida. O segredo dessa auto-organização está na habilidade que os insetos sociais têm de transmitir informações de forma a comunicar outros elementos do grupo sobre o que está sendo feito e influenciar decisões individuais.

Nesse exemplo da construção de um ninho por cupins, a forma de comunicação é feita de maneira indireta, através da secreção de uma substância química denominada *feromônio* (Courtois & Heymans, 1991). Inicialmente, os cupins exploram a área em torno de onde o ninho será construído de forma *cega*. Ao encontrar algum elemento apropriado para a construção do ninho, como terra úmida ou pedaços de madeira, eles carregam tal elemento até a região de construção do ninho. Ao descarregar o elemento, o cupim segrega o feromônio de forma que outros cupins não retirem esse elemento do lugar.

Após essas primeiras passagens, a comunicação indireta começa a emergir dentro da população, pois toda vez que um cupim que carrega um elemento encontra uma pilha de elementos com feromônio depositado, ele tende a depositá-lo nessa pilha proporcionalmente à intensidade do feromônio, exceto quando essa pilha já tenha atingido determinada altura. Essas duas percepções, intensidade do feromônio e altura da pilha, representam duas fontes de informação do processo de decisão efetuado pelo cupim: uma decisão baseada em conhecimento proveniente de outros cupins, ou conhecimento global, e uma decisão baseada em conhecimento próprio, ou local, respectivamente em relação ao feromônio e à altura.

É possível perceber também que o feromônio depositado em uma pilha afeta uma pequena área em torno dela, de forma gradual e que incentive o depósito dos elementos de construção ao lado dela também, gerando uma organização e ordenação na construção do ninho. Partindo de regras tão simples, em Courtois & Heymans (1991) foi possível reproduzir, através de simulações, uma estrutura tão complexa como o ninho, com o passar do tempo.

Apesar desse exemplo dado, outras formas de comunicação são possíveis entre os insetos sociais:

diretas, como a *dança* efetuada pelas abelhas para informar sobre a localização de flores com pólen; e indiretas, como o depósito de feromônio explicado anteriormente. Cabe mencionar aqui que algumas espécies de insetos operam com múltiplos tipos de feromônio, cada qual com a sua funcionalidade. As diversas formas de comunicação podem influenciar um grupo ou a totalidade de indivíduos, e podem influenciar também poucos ou um único indivíduo (Billen, 2006). O termo associado a essa comunicação dentro de grupos sociais de insetos é *estigmergia*, das palavras gregas *stigma* e *ergon*, que em uma tradução livre significam *sinalização do trabalho*.

Muitas das tarefas complexas feitas pelos insetos sociais levam a aplicações diretas em problemas reais de engenharia e computação. Por exemplo, a busca por caminhos mínimos, da mesma forma que esses insetos buscam por fontes de alimentos; agrupamento de dados, inspirando-se na forma com que os insetos limpam o ninho ou estocam alimentos para o inverno; e escalonamento de processos, da mesma forma que alguns insetos conseguem automaticamente delegar tarefas específicas para cada indivíduo, diante de diferentes situações.

Por conta disso, surgiu o interesse no estudo desse comportamento emergente e em como modelar tal comportamento de forma algorítmica, para resolver problemas reais de otimização. Esse estudo foi denominado de *Inteligência de Enxame* por causa da inteligência que emerge da interação de enxames de indivíduos simples de uma colônia.

4.2 A união faz a força

Computacionalmente, o que torna a Inteligência de Enxame atraente para a solução de problemas é o fato de ser um sistema composto por algoritmos simples e computacionalmente eficientes, que podem ser utilizados naturalmente de forma independente – em paralelo – de tal forma que o custo total seja minimizado. Tendo isso em vista, esse tipo de sistema consegue unir, de forma eficiente, pequenas unidades simples de processamento, com custo baixo, gerando uma central com alto poder de processamento.

Além disso, essa independência entre as unidades acrescenta robustez ao sistema, uma vez, que com a falha de um pequeno grupo de unidades, o conjunto total ainda consegue resolver o problema. Utilizando como analogia do exemplo ilustrativo anterior, um cupim que não consiga encontrar material para a construção do ninho não prejudica a conclusão da tarefa, podendo até ajudar a evitar que outros indivíduos explorem a mesma região improdutiva.

Uma outra propriedade que esse sistema possui é a flexibilidade para reagir às mudanças no ambiente. Caso, em determinado momento, esgote-se o material do local onde os cupins estavam retirando material para o ninho, por ser um sistema distribuído e que certamente contém alguns indivíduos explorando caminhos alternativos, a mudança de foco na busca por novos pontos contendo materiais vai

se dar de forma gradativa. Com a mudança do ambiente, os cupins param de depositar o feromônio nesse caminho. Em pouco tempo, ajudada pela evaporação do feromônio já existente, a colônia se auto-regula e se auto-organiza para explotar as fontes alternativas e buscar novas fontes que possam existir, sem a necessidade de alterar o seu funcionamento simples.

Diversas fontes de inspiração compõem o conjunto de sistemas de inteligência de enxame. Dentre eles, os mais conhecidos são: *Sistemas de Formigas*, do inglês *Ant Systems* (AS) (Dorigo, 1992); *Enxame de Partículas* (Kennedy & Eberhart, 1995), do inglês *Particle Swarm* (PS); *Algoritmos inspirados em Colônia de Abelhas* (Pham et al., 2006), do inglês *Bee Algorithms*; *Busca de Difusão Estocástica* (Bishop, 1989), do inglês *Stochastic Diffusion Search* (SDS); *Algoritmo de Busca Gravitacional* (Rashedi et al., 2009), do inglês *Gravitational Search Algorithm* (GSA); *Algoritmo de Gotas d'água Inteligentes* (Shah-Hosseini, 2007), do inglês *Intelligent Water Drops* (IWD).

Em sistemas de Inteligência de Enxame, geralmente são empregadas as seguintes terminologias:

- *agente*: cada entidade individual do sistema, capaz de construir uma solução de forma simples e eficiente;
- *estigmergia*: processo pelo qual os agentes interagem e se comunicam;
- *exploração*: ato de buscar informações ainda não descobertas, ou seja, tender para uma busca dando um peso menor para a informação já adquirida por outros agentes;
- *explotação*: ato de buscar soluções dando um peso maior às informações adquiridas pelos agentes.

4.3 Otimização baseada em colônia de formigas

O algoritmo precursor dos sistemas de Inteligência de Enxame, que teve sua criação em 1992 (Dorigo, 1992), é o algoritmo denominado *Otimização por Colônia de Formigas*. A ideia inicial desse algoritmo surgiu do estudo feito por Goss et al. (1989) sobre o comportamento de formigas argentinas na busca por alimentos. Nesse estudo, foram feitos experimentos em que uma fonte de alimento foi colocada a certa distância do ninho e obstáculos foram criados de tal forma que somente dois caminhos entre o ninho e o alimento eram possíveis.

O caminho superior era mais curto do que o caminho inferior, fazendo com que esse fosse o mais desejável. Inicialmente, Goss et al. (1989) perceberam que as formigas escolhiam o caminho uniformemente, ou seja, com igual probabilidade entre um e outro. Porém, em pouco tempo, a grande maioria das formigas passou a escolher o caminho superior com maior frequência. Contudo,

o caminho inferior ainda era escolhido, mas com menor probabilidade. Esse experimento é ilustrado na Fig. 4.1.

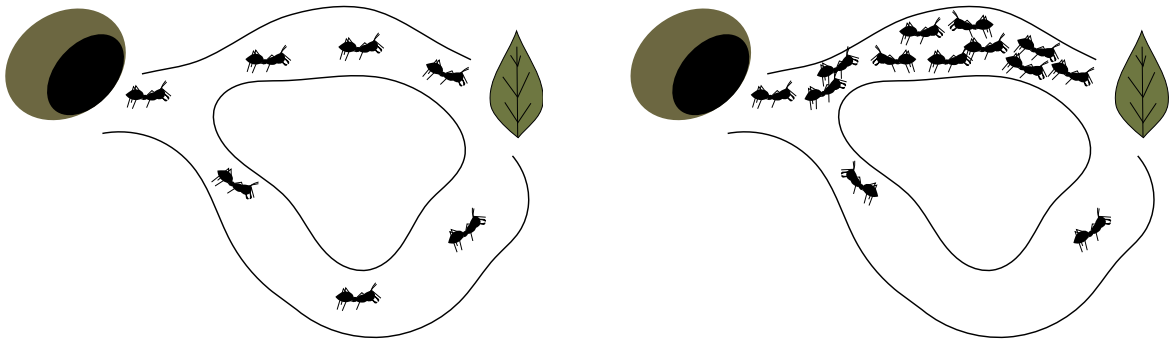


Fig. 4.1: Experimento realizado por Goss et al. (1989) com formigas argentinas. A figura (a) ilustra o instante inicial, onde as formigas exploram os dois caminhos uniformemente. Em (b) é possível ver um instante posterior, onde boa parte das formigas explotam a informação obtida até então.

Nesse experimento, o comportamento que ocorreu de fato foi o mesmo já exemplificado da construção de ninhos pelos cupins: o uso de estigmergia. Em espécies cujos indivíduos são guiados apenas pelo feromônio, quando a formiga sai de seu ninho e parte em busca de alimentos, ela segrega uma trilha de feromônio de forma que ela tenha informação suficiente para traçar o caminho de volta; essas espécies conseguem identificar o caminho de ida e de volta através do ângulo em que o feromônio foi depositado. Já outras espécies podem identificar o caminho de volta através de bússolas naturais ou marcos visuais (Jackson et al., 2004).

Se, após determinado intervalo de tempo, a formiga não encontrar uma fonte de alimentos, ela retorna ao ninho seguindo a trilha de volta, sem depositar feromônio. Caso ela encontre, ela retorna ao ninho depositando feromônio.

Essa regra simples pode levar a três situações:

- A trilha criada pela formiga que não encontrou alimento tende a evaporar mais rapidamente, por não ter o reforço de feromônio no retorno ao ninho.

- A trilha criada pela formiga que encontrou o alimento pelo caminho mais longo tende a obter reforço de feromônio, no retorno ao ninho.

- A trilha criada pela formiga que encontrou o alimento pelo caminho mais curto tende a obter reforço de feromônio, no retorno ao ninho, mais rapidamente e com maior frequência do que no caso do caminho mais longo.

Por essas três situações, é fácil perceber que o caminho que terá maior concentração de feromônio será o caminho mais curto. A concentração de feromônio, da mesma forma que no caso dos cupins, cria um viés no processo de decisão de qual caminho seguir. Com uma tendência maior a seguir pelo caminho mais curto, boa parte das formigas passarão a reforçar com maior frequência este caminho, tornando-o cada vez mais atraente para a decisão de cada formiga. Já o caminho mais longo será decisão de cada vez menos formigas, de tal forma que o feromônio nele diminua gradativamente. Isso torna esse caminho a escolha de poucas formigas, com tendência a explorar novos pontos do espaço de busca.

Diante desses resultados, Dorigo (1992) percebeu que essa tarefa das formigas remete a um problema famoso da área de otimização combinatória: o *Problema do Caixeiro Viajante* (PCV). Esse problema consiste em, dado um grafo completo com N vértices e cada aresta ligando dois vértices tendo um peso definido, encontrar um conjunto de arestas de tal forma que cada vértice do sub-grafo correspondente tenha ligação com apenas dois outros vértices e a somatória dos pesos desse conjunto seja minimizado. Em outras palavras, procura-se pelo menor caminho que passe uma única vez por cada vértice.

A forma como o comportamento da formiga pode ser traduzido para esse problema é bastante simples. Primeiramente, é necessário que cada agente seja capaz de gerar uma solução de forma simplificada. Geralmente, a melhor opção nesse caso é adotar uma estratégia *gulosa*, ou seja, inicia-se em um determinado vértice e, a cada passo, escolhe-se o vértice cujo peso da aresta ligando este ao vértice atual seja o menor possível, dentre os vértices que ainda não foram escolhidos. Em sistemas de Inteligência de Enxame, qualquer informação que é inerente ao problema e não é alterada durante o procedimento é denominada de informação heurística.

Uma vez que essa estratégia é determinística e um agente trabalha de forma estocástica, pode-se alterar essa estratégia de forma que a escolha do próximo vértice seja aleatória, com probabilidade inversamente proporcional ao peso da aresta, ou seja, quanto menor o peso maior a chance de ser escolhida. O próximo passo do algoritmo é a criação de uma forma de comunicação indireta. Para tanto, usa-se uma matriz, denominada de *matriz de feromônio*, onde cada elemento (i, j) reflete a qualidade global da aresta que interliga os vértices i e j dentro de uma solução. Os valores dessa matriz são atualizados toda vez que um agente completa uma solução. A atualização pode se dar por incremento ou decremento dos valores, de acordo com a qualidade da solução obtida.

Com essa matriz de feromônio, as próximas iterações de cada agente podem então utilizar seus valores para construir uma nova solução, em vez de utilizar os valores dos pesos das arestas. Alternativamente, pode-se usar a combinação dessas duas informações, heurística e feromônio, para servir como decisão na construção da solução.

Essencialmente, esse algoritmo cria uma tabela contendo a informação da probabilidade de dada

aresta pertencer à solução ótima, dado o conjunto de amostras geradas até o momento. O algoritmo básico do ACO está ilustrado em Alg. 5 e seus procedimentos internos serão explicados a seguir.

Algoritmo 5 Algoritmo básico de Otimização por Colônia de Formigas.

```
enquanto não_convergir faça
    para cada formiga faça
        construir_solução()
    fim para
    atualizar_feromônio()
fim enquanto
```

Conforme pode ser visto no Alg. 5, o ACO é uma meta-heurística bem simples que consiste na execução de três procedimentos: um critério de convergência do algoritmo; uma função para construção da solução, levando em conta as informações inerentes do problema e aquelas adquiridas pelos agentes; e uma função para atualizar a informação dos agentes e cumprir o papel de comunicação indireta.

4.3.1 Construindo Soluções

Conforme dito anteriormente, a construção de uma solução é feita incrementalmente escolhendo um vértice a cada passo do procedimento, até que se obtenha uma solução completa. A escolha de cada vértice em um dado passo da construção é feita de forma estocástica, seguindo a probabilidade dada por:

$$p_{i,j}^{k}(t) = \begin{cases} \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum_{l \in J^k} \tau_{i,l}^{\alpha} \cdot \eta_{i,l}^{\beta}}, & \text{se } j \in J^k \\ 0, & \text{c.c.} \end{cases} . \tag{4.1}$$

Essa equação pode ser interpretada como a probabilidade do agente k escolher o vértice j como o próximo elemento da solução, dado que a última escolha foi o vértice i, na iteração t, dentro do conjunto J^k dos vértices que ainda são factíveis para a solução. Para isso, é utilizada a combinação normalizada de $\tau_{i,j}$, o valor de feromônio depositado na aresta (i, j) e $\eta_{i,j}$, a informação heurística dessa aresta, combinados com peso α e β, respectivamente.

Para o caso do PCV, o conjunto de vértices factíveis são todos aqueles que ainda não fazem parte da solução. Já a informação heurística, conforme dito anteriormente, pode ser tomada como o inverso da distância entre os dois vértices, ou $\frac{1}{d_{i,j}}$. A Fig. 4.2 ilustra passo a passo a construção de uma solução por um agente dentro do ACO para o PCV.

Uma vez que cada agente construiu uma solução, é hora de capturar as informações obtidas pelas amostras geradas, de forma que a matriz de feromônio contenha uma informação mais precisa do

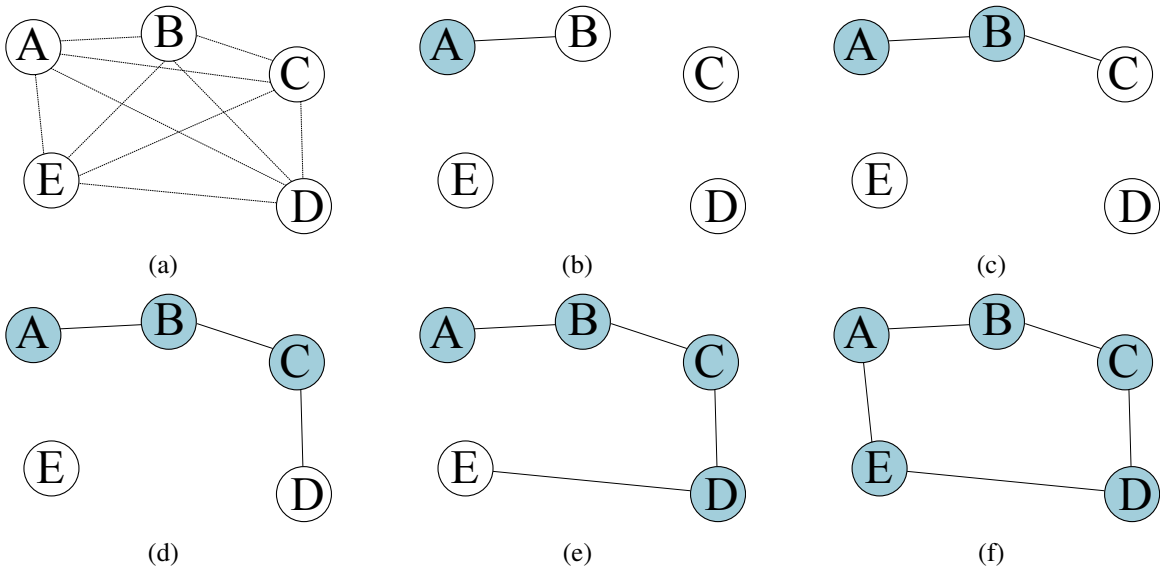


Fig. 4.2: Construção de uma solução de PCV para o grafo ilustrado em (a). Em (b) escolhe-se o vértice A como passo inicial e, de acordo com o peso das arestas, o vértice B é escolhido como o próximo passo, incluindo a aresta (A, B) na solução. Sequencialmente, os passos (c), (d), (e) e (f) repetem esse procedimento, incluindo as arestas correspondentes. Ao final da operação, obtém-se um ciclo completo.

que aquela apresentada na iteração anterior. Isso pode ser feito de forma assíncrona, ou seja, assim que um agente completa a solução, as informações de feromônio são imediatamente atualizadas; ou síncrona, quando o algoritmo aguarda até que todos os agentes retornem suas soluções para, então, atualizar.

A atualização dos valores da matriz de feromônio é feita através da qualidade das amostras geradas pelos agentes em uma dada iteração, e calculada de forma incremental como segue:

$$\tau_{i,j}(t+1) = (1-\rho) \cdot \tau_{i,j}(t) + \rho \cdot \Delta\tau_{i,j}, \tag{4.2}$$

onde ρ é a taxa de *evaporação* do feromônio, utilizada para evitar a convergência prematura dos valores da matriz de feromônio e limitada dentro da faixa de valores $[0, 1)$, e $\Delta\tau_{i,j}$ é o valor que será adicionado ao elemento (i, j) da matriz de feromônio, calculado de forma proporcional à qualidade da solução gerada pelo agente. No caso do PCV, a qualidade da solução é calculada como a somatória dos pesos das arestas e como o objetivo é minimizar essa função, o valor de incremento será:

$$\Delta\tau_{i,j} = \begin{cases} \sum_{k \in K} \frac{1}{f(S_k)}, & \text{se } (i,j) \in S_k \\ 0, & \text{c.c.} \end{cases}, \tag{4.3}$$

onde $f(S_k)$ é o valor da função-objetivo do problema para a solução S_k gerada pela formiga k. Note

que essa fórmula generaliza outras formas de se calcular esse incremento. Em algumas variantes do ACO, ele é calculado apenas pelo conjunto K de uma porcentagem das melhores formigas, ou K é um conjunto unitário com a melhor solução encontrada até o momento.

Com essa abordagem simples e descentralizada, o ACO obteve um desempenho de destaque, não só no tratamento de instâncias do PCV mas em diversas outras aplicações de otimização combinatória. Entretanto, diversas modificações e extensões foram feitas na estrutura original de forma a obter melhorias na qualidade das soluções obtidas. Três delas serão descritas a seguir.

4.3.2 Max-Min Ant System

Embora almejando simular as propriedades observadas em colônias de insetos, como robustez, flexibilidade e auto-organização, o ACO original obteve sucesso parcial em mantê-las de forma coerente durante todo o processo. Exemplificando com a flexibilidade, dado um número suficiente de iterações em que o ambiente não se altera, o nível de feromônio nas arestas da melhor solução tende a atingir valores exageradamente altos. Isso cria um viés muito grande para essa solução, impossibilitando a descoberta de novas regiões interessantes no espaço de busca. Esse problema é comum em algoritmos populacionais e é referenciado como *convergência prematura*.

Esse problema ocorre principalmente pela existência de ótimos locais com qualidade muito maior que outros pontos do espaço de busca. Ao encontrar esses pontos, a probabilidade calculada tende a favorecer tal ótimo local em detrimento de outras áreas ainda não exploradas. Em poucas iterações, o viés gerado por essa solução faz com que nenhuma outra região seja explorada. De forma a minimizar esse problema, Stützle & Hoos (2000) propuseram algumas funcionalidades novas e restrições ao procedimento do ACO original, criando o algoritmo denominado *Max-Min Ant System* (MMAS).

As alterações feitas nesse algoritmo são referentes à atualização dos níveis de feromônio. A primeira alteração é na escolha de qual conjunto de soluções será utilizado para o cálculo do incremento na Eq. 4.3. No MMAS, esse cálculo é feito apenas com duas soluções, a melhor solução global, ou seja, a melhor encontrada até o momento entre todas as iterações, e a melhor solução local, dada pela melhor solução dentre as encontradas na iteração atual. Esse procedimento incentiva a manutenção da melhor solução encontrada e, ao mesmo tempo, tenta direcionar para novas possibilidades através da solução local. Isso evita também algo que estimulava a convergência prematura no ACO original, quando em uma dada iteração eram geradas diversas soluções iguais, aumentando significativamente a influência daquela solução.

A outra alteração, que deu origem ao nome desse método, impõe limites máximo e mínimo aos valores do feromônio. Ao aplicar a Eq. 4.2, se o novo valor calculado do feromônio for menor que τ_{min} ou maior que τ_{max}, os valores são substituídos por esses limitantes, de acordo com qual deles foi desrespeitado. Com essa limitação, mesmo que uma solução ótima local seja encontrada, ela

não poderá impactar nos níveis de feromônio além de determinado valor, limitando o máximo de influência que ela poderá exercer na Eq. 4.1.

Conforme discutido em Stützle & Dorigo (1999) e Stützle & Hoos (2000), o parâmetro τ_{min} tem maior influência que τ_{max} na convergência do algoritmo. O parâmetro τ_{max} pode ser calculado como o valor máximo esperado em uma situação na qual a solução ótima global é sempre obtida pelo algoritmo durante infinitas iterações. Partindo da Eq. 4.2, referente à atualização dos valores do feromônio, e dado que uma das formigas no algoritmo sempre construirá a solução ótima S_{opt}, o feromônio de uma aresta $(i, j) \in S_{opt}$ na iteração t será:

$$\tau_{i,j}(t) = (1-\rho)^t \cdot \tau_0(t) + \rho \cdot \frac{1}{f(S_{opt})} \sum_{t'=1}^{t} (1-\rho)^{t-t'}, \tag{4.4}$$

onde τ_0 é o valor inicial dos elementos da matriz de feromônio e sem levar em conta uma possível informação heurística. Uma vez que o valor de ρ é limitado entre $[0, 1)$ e com $t \to \infty$, obtém-se:

$$\lim_{t\to\infty} \tau_{i,j}(t) = \lim_{t\to\infty} (1-\rho)^t \cdot \tau_0(t) + \lim_{t\to\infty} \rho \cdot \frac{1}{f(S_{opt})} \sum_{t'=1}^{t} (1-\rho)^{t-t'}, \tag{4.5}$$

$$\lim_{t\to\infty} (1-\rho)^t \cdot \tau_0(t) = 0, \tag{4.6}$$

$$\lim_{t\to\infty} \rho \cdot \frac{1}{f(S_{opt})} \sum_{t'=1}^{t} (1-\rho)^{t-t'} = \rho \cdot \frac{1}{f(S_{opt})} \cdot \frac{1}{\rho} = \frac{1}{f(S_{opt})}, \tag{4.7}$$

$$\tau_{max} = \lim_{t\to\infty} \tau_{i,j}(t) = \frac{1}{f(S_{opt})}. \tag{4.8}$$

Em situações práticas, não é possível determinar o valor de $f(S_{opt})$. Porém, em alguns casos é possível encontrar um limitante superior/inferior e, no pior dos casos, calcular uma estimativa acrescida de um valor de tolerância. Outra forma aproximada de substituir esse valor é utilizar o valor da melhor solução encontrada até o presente momento.

O valor de τ_{min} é calculado de forma que ele possa ser determinado a partir do valor de probabilidade da construção da melhor solução obtida até o momento. Com isso, é possível determinar com maior segurança o grau máximo de viés introduzido nas probabilidades das soluções. Dado que p_{pert} é a probabilidade de uma dada aresta pertencente a S_{opt} ser escolhida em cada passo da construção da solução, a probabilidade p_{opt} da solução S_{opt}, de tamanho n, ser construída por um agente é dada por:

$$p_{opt} = p_{pert}^{n-1} = (e^{\frac{\ln p_{opt}}{n-1}})^{n-1}. \tag{4.9}$$

Com uma média de $\frac{n}{2}$ vértices candidatos a serem escolhidos em cada passo da construção do

algoritmo, através da Eq. 4.9, p_{pert} pode ser aproximado como:

$$p_{pert} = \frac{\tau_{max}}{\tau_{max} + \frac{n}{2} \cdot \tau_{min}}. \tag{4.10}$$

Com isso, é possível determinar o valor de τ_{min} de forma a decidir a probabilidade máxima da escolha de uma aresta pertencente à solução ótima. Em Stützle & Hoos (2000), foi sugerida uma probabilidade de 80%, que leva a uma proporção de $\frac{\tau_{max}}{\tau_{min}} = 2n$. Na execução do algoritmo, foi introduzido um passo adicional de busca local aplicada a cada solução construída. Porém, a atualização do feromônio é feita levando-se em conta as arestas das soluções originais.

4.3.3 ACO Hipercubo

Apesar de efetivamente melhorar a flexibilidade, o MMAS ainda apresentava dificuldades quando enfrentando problemas com dimensão elevada. Em Blum & Dorigo (2004), novas complementações foram propostas ao ACO para aumentar ainda mais as propriedades inerentes de um sistema de Inteligência de Enxame nesse algoritmo. A proposta, denominada *estrutura hipercubo para o ACO*, do inglês *ACO HyperCube Framework* (HCF-ACO), fixou os limites mínimo e máximo dos valores de feromônio pelo valor normalizado $(0, 1)$. Para tanto, a Eq. 4.3 utiliza o valor normalizado referente à qualidade da solução da seguinte maneira:

$$\Delta\tau_{i,j} = \begin{cases} \sum_{k \in K} \frac{F(S_k)}{\sum_{k' \in K} F(S_{k'})}, & \text{se } (i, j) \in S_k \\ 0, & \text{c.c.} \end{cases}, \tag{4.11}$$

onde $F(S_k)$ é uma função que mede a qualidade objetivando a maximização. Além disso a matriz de feromônio teve seus valores iniciados em $0,5$. Essas modificações fizeram com que a diferença absoluta entre a pior e a melhor solução encontrada fosse reduzida a ponto de, apesar de um agente preferir explorar arestas da melhor solução, a probabilidade de desviar do caminho ótimo é alta o suficiente para evitar a convergência.

4.3.4 MMAS Melhorado

Percebendo alguns problemas ainda existentes no HCF-ACO, em de França et al. (2004a,b, 2005) novas alterações foram propostas para melhorar os resultados. O primeiro ponto observado pelos autores foi na Eq. 4.11, que assumia o valor zero como mínimo da função de avaliação. Com essa suposição, os valores não eram normalizados corretamente e ocorria erro na escala dos valores. Em problemas em que a diferença do valor mínimo real ao valor assumido era grande, ainda existia ocorrência de convergência prematura.

Diante disso, e como os limitantes inferiores e superiores da função-objetivo são desconhecidos, a Eq. 4.11 foi modificada da seguinte forma:

$$\Delta\tau_{i,j} = \begin{cases} \sum_{k \in K} \frac{F(S_k) - F(S_{pior})}{F(S_{melhor}) - F(S_{pior})}, & \text{se } (i,j) \in S_k \\ 0, & \text{c.c.} \end{cases}, \tag{4.12}$$

onde S_{melhor} e S_{pior} são a melhor e a pior soluções encontradas até o momento. Essa equação distribui de uma forma mais uniforme o incremento de feromônio pelas soluções geradas.

Outra alteração proposta para evitar a convergência prematura do sistema foi a detecção de convergência. A convergência ocorre quando, dado que cada solução gerada tem tamanho fixo n, caso n arestas atinjam o limitante superior dos valores de feromônio enquanto o restante permaneça no limitante inferior, então apenas uma solução completa contém valores máximos. Com isso, para detectar o momento de convergência do algoritmo, são utilizadas as seguintes equações:

$$\sum_{\tau_{i,j} \geq \tau_{max} - \sigma} 1 = n, \tag{4.13}$$

$$\sum_{\tau_{i,j} \leq \tau_{min} + \sigma} 1 = |E| - n, \tag{4.14}$$

onde σ é um limiar de imprecisão e $|E|$ é o número de arestas. Uma vez que o sistema encontra-se nessa situação, o valor inicial τ_0 é atribuído a todos os elementos da matriz de feromônio, voltando para a condição inicial do sistema, com probabilidade uniforme.

Apesar desse reinício do procedimento, os valores atuais de $F(S_{melhor})$ e $F(S_{pior})$ são mantidos, fazendo com que os valores calculados pela Eq. 4.12 sejam menores do que na condição inicial. Isso torna a busca mais *cuidadosa* em relação a outras regiões, até que uma nova convergência ocorra.

4.4 Heurística construtiva para δ-biclusterização

Conforme visto nas seções anteriores, para criar um algoritmo utilizando Inteligência de Enxame é desejável encontrar uma heurística construtiva capaz de gerar soluções razoáveis, com custo computacional baixo, e que permita elementos probabilísticos. No Cap. 3, foi apresentada uma heurística construtiva para a δ-Biclusterização, denominada CC. Nessa heurística, a construção de um bicluster partia da base completa e removia iterativamente as linhas e colunas cujo resíduo excedia o limiar. Após esse procedimento, algumas linhas e colunas eram inseridas novamente se, nessa ocasião, possuíssem um valor residual menor ou igual ao resíduo atual do bicluster.

Para encontrar mais do que um bicluster, a cada término desse procedimento os elementos do

bicluster anterior eram mascarados com valores aleatórios de forma a aumentar o ruído e incentivar o algoritmo a procurar em áreas ainda não exploradas. Porém, esse procedimento gera dois problemas ao ser aplicado em um sistema de Inteligência de Enxame. O primeiro problema é que a busca de um bicluster depende dos biclusters obtidos anteriormente, pois existe a necessidade de inserir ruído de forma a aumentar a diversidade. Isso impossibilita a busca em paralelo que os agentes de um enxame possuem de forma natural. O outro problema está diretamente relacionado à introdução de ruído, pois, com o passar das iterações e com a base de dados cada vez com mais ruído, a qualidade das soluções encontradas pelo algoritmo se torna gradativamente pior a cada passo.

De forma a reduzir o custo computacional e permitir a paralelização dos agentes, é proposta aqui uma nova heurística construtiva para a busca de δ-biclusters (de França & Von Zuben, 2010). Esse procedimento é capaz de, isoladamente, encontrar, em média, bons biclusters e também biclusters direcionados para regiões desejadas (ainda não exploradas), sem a necessidade de gerar ruído junto aos elementos da base de dados. Tal procedimento está descrito no Alg. 6 e será detalhado na sequência.

Algoritmo 6 Heurística construtiva para encontrar um δ-bicluster.

Inicie o bicluster B com uma linha i escolhida aleatoriamente e o conjunto C de todas as colunas da base de dados;
para cada linha da base de dados, exceto a linha i **faça**
  Insira a linha no bicluster B;
  Calcule o RQM de cada coluna segundo a Eq. 3.22;
  Remova todas as colunas que possuírem RQM maior do que δ;
  se o bicluster resultante possuir um número de colunas inferior a col_{min} **então**;
    Volte ao estado anterior e adicione a mesma linha com valores negativos;
    Calcule o RQM de cada coluna, segundo a Eq. 3.22;
    Remova todas as colunas que possuírem RQM maior do que δ;
    se o bicluster resultante possuir um número de colunas inferior a col_{min} **então**;
      Volte ao estado original;
    fim se
  fim se
  se o bicluster resultante possuir exatamente col_{min} colunas **então**
    Saia do laço
  fim se
fim para
Execute o procedimento de inserção de múltiplos nós (Alg. 3) em B.

A heurística construtiva inicia com um bicluster de tamanho reduzido contendo apenas uma linha e o conjunto de todas as colunas da base de dados. A linha inicial pode ser escolhida aleatoriamente ou deterministicamente, conforme o cenário da aplicação. Como exemplo, quando uma região da base de dados é de particular interesse, pode-se iniciar o procedimento apenas nas linhas que contém tal região. Note que aqui será utilizada a terminologia *linha* e *coluna* para generalizar o procedi-

mento, uma vez que esse procedimento pode ser aplicado na base de dados transposta, invertendo o significado de objetos e atributos. Esse aspecto inicial será importante para a aplicação no contexto de Inteligência de Enxame, conforme será explicado na próxima seção.

Uma vez que se escolhe a linha e se obtém um bicluster reduzido, é necessário aumentar seu volume inserindo novas linhas. Para isso, é necessário primeiro escolher a ordem em que essas inserções devem ser feitas. Essa ordem pode ser sequencial, seguindo sua ordem original, ou seguindo uma ordem aleatória, ou então determinada por alguma regra. Definida a ordem, adiciona-se uma linha de cada vez e, ao adicionar, verifica-se o valor do RQM para cada coluna. As colunas com um valor residual maior do que o estabelecido por δ são removidas. Se o bicluster resultante dessa operação possuir um número de colunas menor do que um limiar estabelecido, a adição dessa linha é desfeita e o conjunto de colunas anterior é restaurado. Com isso, garante-se que o bicluster final não contenha um volume muito pequeno e, consequentemente, de pouca valia para a análise da base de dados. Se esse for o caso, pode-se tentar adicionar a mesma linha, porém com valores negativos, assim como descrito no Alg. 3 do método CC, repetindo em seguida o procedimento de remoção de colunas. Caso ainda assim o número de colunas seja menor que o limiar, a operação é novamente desfeita e o procedimento passa para a próxima iteração.

Note que a ordem com que as novas linhas são adicionadas irá influenciar na qualidade do bicluster final, pois, dependendo da escolha, um conjunto diferente de colunas será removido, o que interfere na adição de mais ou menos linhas nas próximas iterações. Caso o número mínimo de colunas tenha sido atingido ou então todas as linhas tenham sido verificadas, efetua-se uma busca local, da mesma forma que o algoritmo de Inserção de Múltiplos Nós do CC, para aumentar o volume do bicluster, se possível. Esse procedimento pode se tornar necessário, pois uma linha que não era apta a ser adicionada ao bicluster, em determinado momento, pode se tornar apta com o conjunto de colunas do bicluster final. Da mesma forma, uma coluna que foi removida, em certo momento, pode se tornar apta a ser adicionada na configuração final. A saída do laço por causa do limite mínimo de colunas ocorre porque, ao atingir esse mínimo, nenhuma operação de remoção de colunas poderá ser feita e, portanto, a busca local é suficiente para inserir as linhas restantes.

Da mesma forma que a ordem das linhas influencia na qualidade final do bicluster, a escolha inicial influenciará na região do espaço de busca que será explorada. Com isso, partindo de uma linha inicial diferente, tende-se a explorar regiões diferentes do espaço de busca. Esse procedimento, por ser construtivo, ou seja, por partir de um bicluster inicial reduzido e iterativamente incrementar esse bicluster, tende a encontrar biclusters de forma mais eficiente, em termos computacionais, do que o algoritmo CC. A próxima seção explicará como utilizar essa heurística no contexto de ACO e Inteligência de Enxame, visando obter um conjunto de biclusters que maximize a cobertura da base de dados.

4.5 δ-biclusterização com inteligência de enxame: SwarmBcluster

Conforme mencionado na seção anterior, a heurística construtiva descrita no Alg. 6 tem sua qualidade influenciada pela ordem em que as linhas são inseridas em um bicluster. Dessa forma, a tarefa de encontrar o bicluster coerente com volume máximo, partindo de um determinado bicluster inicial, é equivalente a encontrar a melhor ordenação de inserção de linhas para esse bicluster. Encontrar a ordem que obtenha aproveitamento máximo em um conjunto de nós, sem repetições, é um problema semelhante ao PCV, descrito anteriormente.

Bastaria então aplicar o algoritmo ACO utilizando essa heurística construtiva. A ordenação é probabilística e proporcional ao feromônio referente à ordem da linha atual e à próxima linha a ser inserida. O restante do procedimento é feito de forma igual ao algoritmo MMAS Melhorado, com exceção do procedimento de detecção de convergência. Essa adaptação terá uma forma própria de lidar com a diversidade de soluções. Outro aspecto que deve ser definido é a função de avaliação a ser utilizada para atualização de feromônio, a qual será proporcional ao volume do bicluster. Além do objetivo explícito associado ao volume, outros objetivos também serão otimizados, sendo eles a minimização do RQM, controlado pela restrição δ, a minimização da sobreposição e a maximização da cobertura. A forma como tratar desses dois últimos objetivos será discutida posteriormente.

Opcionalmente, uma informação heurística também pode ser utilizada para melhorar a tomada de decisão de quais linhas inserir. Nesse caso, será utilizado o inverso da distância euclidiana entre os objetos da base de dados. Essa escolha foi feita de forma a ter uma informação global acompanhando as informações do feromônio, e que seja de cálculo simples e eficiente.

Até então, essa adaptação de ACO é capaz de retornar apenas um único bicluster, de forma similar ao realizado pelo algoritmo CC. Porém, dada a forma como a heurística construtiva é efetuada, existe um modo mais elegante de encontrar um conjunto de biclusters. Conforme dito anteriormente, o ponto inicial da heurística construtiva influencia na determinação da região onde o bicluster será explorado. Portanto, para encontrar um bicluster em uma região diferente bastaria aplicar o ACO com um ponto inicial distinto do anterior. Porém, deseja-se obter um conjunto de biclusters com sobreposição mínima e, caso a nova linha inicial esteja contida no bicluster gerado anteriormente, intuitivamente as chances de encontrar um bicluster que tenha alta sobreposição em relação ao anterior são altas.

Com isso, é interessante determinar como ponto inicial do algoritmo apenas as linhas da base de dados que não pertençam a nenhum bicluster até o momento. Seguindo esse procedimento, garantir-se-ia também que todas as linhas da base de dados estariam presentes dentro de algum bicluster, maximizando também a cobertura. Embora esse procedimento resolva a questão da cobertura de

linhas, isso não garante a cobertura geral da base de dados, uma vez que essa regra de decisão ignora as colunas que não foram cobertas em determinadas linhas. Isso ocorre pois um determinado bicluster pode conter a linha i e o conjunto J de colunas; portanto, não haverá novas tentativas de iniciar um bicluster com a linha i. Ao final do procedimento, se nenhum dos biclusters, além daquele, contiver a linha i, a área da base de dados que compreende essa linha com o conjunto C das colunas não pertencentes ao conjunto J não será coberta.

Para resolver isso, cria-se então uma *lista de candidatos* que representa a lista de linhas que ainda não foram totalmente cobertas até o momento. Cada elemento dessa lista contém uma outra lista relatando todas as colunas que ainda não foram cobertas por nenhum bicluster nessa linha. A Fig. 4.3 ilustra a estrutura dessa lista. Vale notar que o tamanho das listas de cada linha é variável, pois depende dos elementos não cobertos ainda, conforme é possível perceber na ilustração.

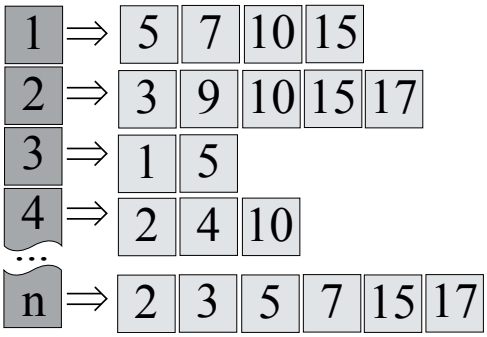


Fig. 4.3: Lista de candidatos representando as colunas ainda não cobertas pelos biclusters já produzidos, tendo como referência cada linha da base de dados.

Essa lista inicia contendo todas as linhas e, para cada linha, uma lista com todas as colunas. Ao iniciar o algoritmo ACO, a escolha da linha inicial na formação do bicluster é feita aleatoriamente. Concluindo o procedimento, essa lista é atualizada de tal forma que, para todas as linhas pertencentes ao bicluster, removem-se todas as colunas contidas nele (Alg. 7). Quando uma busca por um novo bicluster for iniciada, a decisão da escolha da linha inicial se baseará apenas no conjunto de linhas que tenham o maior número de colunas na lista, e o bicluster inicial será formado por essa linha e seu respectivo conjunto de colunas. Isso deve ser feito para incentivar um aumento da cobertura, uma vez que, ao iniciar com o conjunto completo de colunas, o algoritmo pode retornar um bicluster com poucas colunas da lista de candidatos.

Com esse procedimento, pode-se garantir que o bicluster resultante conterá uma região ainda não explorada da base de dados. Apesar de ser possível maximizar a cobertura dessa forma, algumas regiões podem continuar descobertas quando sobrarem apenas linhas com poucas colunas na lista de candidatos. O procedimento completo, denominado *SwarmBcluster*, do inglês *Swarm Intelligence*

Biclusters, é ilustrado no Alg. 8.

Algoritmo 7 Procedimento para atualizar a lista de candidatos.

para cada linha *i* do bicluster **faça**
  para cada coluna *j* do bicluster **faça**
    Elimine a coluna *j* da entrada correspondente à linha *i* da lista de candidatos;
  fim para
  se lista de colunas para a linha *i* estiver vazia **então**
    Elimine linha *i* da lista de candidatos
  fim se
fim para

Algoritmo 8 Algoritmo SwarmBcluster para buscar um conjunto de δ-biclusters que maximize a cobertura da base de dados, com restrição de sobreposição.

Inicie lista de candidatos;
Inicie lista de biclusters;
enquanto lista de candidatos contiver elementos **faça**
  Escolha o próximo candidato a ser utilizado como ponto inicial;
  Execute procedimento de ACO utilizando heurística descrita no Alg. 6 com esse ponto inicial;
  Verifique sobreposição entre o bicluster resultante e aqueles pertencentes à lista de biclusters;
  se a sobreposição com qualquer um dos biclusters for menor do que *limiar_sobrep* **então**
    Insira bicluster na lista;
  fim se
  Atualize lista de candidatos (Alg. 7);
fim enquanto

Nesse algoritmo, foi acrescentada uma restrição proibindo que dois ou mais biclusters se sobreponham em uma taxa superior a um determinado limiar, combatendo assim a redundância. Caso necessário, é possível também incluir outras restrições dentro desse algoritmo, assim como procedimentos para tratar tais restrições, sem precisar alterar a estrutura interna do mesmo. Exemplificando, ao encontrar dois biclusters que se sobreponham acima do limiar permitido, poderia ser feito um procedimento que unifique os dois biclusters, criando um único bicluster a ser inserido na lista.

Vale notar que esse algoritmo, embora tenha sido inspirado nos conceitos de inteligência de enxame, contém elementos de memória que também são explorados em *Busca Tabu* (Glover & Marti, 2006). A *Busca Tabu* mantém uma lista de soluções proibidas de entrar na solução durante os próximos passos da busca local, evitando, ou postergando, a convergência para um ótimo local. No *SwarmBcluster*, a lista contém as soluções que devem ser utilizadas nos próximos passos, incentivando a exploração de toda a base de dados. De forma equivalente, a construção de várias soluções com um ponto inicial distinto a cada passo remete ao algoritmo conhecido como *GRASP* (Feo &

Resende, 1989), que resolve um mesmo problema partindo de soluções iniciais distintas procurando por aquele que leve à convergência para um ótimo global.

4.6 Metodologia experimental

De forma a avaliar corretamente o desempenho computacional e a qualidade dos resultados obtidos pelo algoritmo SwarmBcluster, assim como traçar um comparativo com outras metodologias, foram feitos experimentos com duas bases de dados já utilizadas na literatura e com um número de repetições suficiente para calcular as quantidades estatísticas. Os experimentos, que serão detalhados a seguir, são similares aos realizados por Coelho et al. (2008).

Nesses experimentos, seis metodologias já explicadas no capítulo anterior foram utilizadas para a comparação de desempenho. São elas:

- algoritmo de Cheng & Church (2000), heurística simples e precursora da δ-biclusterização;
- BIC-aiNet (de Castro et al., 2007a) e MOM-aiNet (Coelho et al., 2008), meta-heurísticas imuno-inspiradas que otimizam múltiplos objetivos do problema de biclusterização, com combinação de objetivos e com abordagem multi-objetivo, respectivamente;
- algoritmo de Mitra & Banka (2006), uma meta-heurística multi-objetivo adaptada de um algoritmo bem conhecido de computação evolutiva e aqui denominada como MOEABIC; e
- bicACO (de França et al., 2008), uma outra meta-heurística baseada no algoritmo ACO.

Duas bases de dados frequentemente utilizadas para avaliar algoritmos de biclusterização foram utilizadas neste trabalho para medir a qualidade das soluções: as bases de expressão gênica conhecidas como *Yeast Genome Dataset* e *Human B-Cell Lymphoma*. Ambas podem ser obtidas no seguinte endereço da internet: `http://sites.google.com/site/fabricioolivetti/datasets`.

A base de dados *Yeast Genome Dataset*, originalmente chamada de *Saccharomyces cerevisiae*, utilizada inicialmente em Cho et al. (1998), contém 2.884 genes (objetos) com 17 condições (atributos) e, para efeitos práticos, teve seus valores faltantes substituídos por valores aleatórios na faixa de $[0, 600]$ por Cheng & Church (2000). Tem-se por objetivo desse experimento encontrar 300 biclusters dentro dessa base de dados, com $\delta \leq 300$ (Cheng & Church, 2000).

A base de dados *Human B-Cell Lymphoma*, introduzida em Alizadeh et al. (2000), possui 4.026 genes sob 96 condições e, assim como no caso do *Yeast*, teve seus valores faltantes substituídos por números aleatórios, porém agora na faixa de $[-750, 650]$. Para essa base, o objetivo é encontrar 200 biclusters com $\delta \leq 1200$ (Cheng & Church, 2000).

Note que esses valores de δ foram determinados por Cheng & Church (2000), baseados nos estudos de clusters nessas duas bases feita em Tavazoie et al. (1999).

A qualidade do conjunto de biclusters foi avaliada através da média, desvio-padrão, valores máximos e mínimos, e mediana, isso para RQM, volume e sobreposição do conjunto. Também foi medido o percentual de cobertura do conjunto de biclusters em relação à base de dados. Para reduzir o efeito do erro estatístico, cada experimento foi realizado 30 vezes.

Os parâmetros para cada algoritmo foram determinados experimentalmente em Coelho et al. (2008), de forma a permitir o melhor desempenho em cada base de dados. Levando em conta o critério de parada, para a BIC-aiNet e MOM-aiNet foram feitas 2.000 iterações para obter o resultado final. Em todos os outros algoritmos, o número de iterações foi igual ao número de biclusters a encontrar.

Relativo ao algoritmo SwarmBcluster, os parâmetros determinados experimentalmente foram $min_{col} = 7$ e $min_{col} = 20$, respectivamente para Yeast e Humans. A taxa permitida de sobreposição foi determinada como 60%, visando dar um espaçamento suficiente entre os biclusters de forma a maximizar a cobertura com o número pré-determinado de biclusters. Quanto ao número de iterações e número de formigas para o algoritmo ACO, em cada iteração, foram utilizadas duas combinações distintas de parâmetros para verificar o desempenho tanto da qualidade dos resultados como do tempo computacional. São eles: $\{\text{iterações, formigas}\} = \{\{5, 5\}, \{15, 15\}\}$, que serão denominados como $SwarmBcluster_5$ e $SwarmBcluster_{15}$, respectivamente. Adicionalmente, foi utilizado o valor $\rho = 0,2$ para a taxa de evaporação de feromônio. Os parâmetros de peso para o feromônio e informação heurística foram $\alpha = 1$ e $\beta = 3$, respectivamente.

O algoritmo BIC-aiNet tem como parâmetros o grau de relevância das linhas e das colunas, aos quais foram atribuídas com os valores de 2 e 3, respectivamente, para Yeast e, 3 e 2, respectivamente, para Humans. Uma vez que o uso de δ nesse algoritmo é em forma de peso para as funções-objetivo, ele não traz garantias de que o bicluster terá um valor abaixo desse limiar. Portanto, o valor desse parâmetro para o Yeast foi escolhido experimentalmente como 100, e para o Human como 200.

Para a MOM-aiNet, o tamanho das sub-populações foi determinado de forma a equilibrar a qualidade obtida com o desempenho computacional. Para isso, foi escolhido um valor de 6 indivíduos por população. Para o parâmetro δ, por conta da estrutura que envolve múltiplos objetivos a serem atendidos simultaneamente, foram utilizados os mesmos valores de δ atribuídos para cada base de dados.

Para o algoritmo MOEABIC, foi determinado um número de 50 soluções-filhas a cada iteração e probabilidades de cruzamento e mutação de 75% e 3%, respectivamente. O valor de α foi escolhido como $1,2$ para Yeast e $1,8$ para Human.

O método de Cheng & Church tem apenas um único parâmetro, além do limiar δ: é o valor α

utilizado na primeira etapa do procedimento. Para ambas as bases foi utilizado o valor de $1,2$.

Finalmente, o algoritmo bicACO utilizou apenas 10 iterações com o número de formigas igual ao número de biclusters requisitado. Uma vez que esse algoritmo parte da base completa e remove as linhas e colunas até atingir o valor exato de δ, esse parâmetro foi experimentalmente determinado como 180 para Yeast. A taxa de evaporação do feromônio foi igual a $0,2$. Por conta de seu alto custo computacional, os experimentos com esse algoritmo na base de dados *Human* não foram realizados.

Todos os experimentos foram realizados em um computador *Intel Core2Quad Q9550 @ 2.83GHz* com *2 GB* de *RAM*, utilizando apenas um único *core* para cada experimento.

4.7 Resultados experimentais

Em cada repetição do experimento, foram mensuradas as quantidades estatísticas de média aritmética, desvio padrão, mediana, valor mínimo e valor máximo calculados sobre o conjunto de biclusters gerados. Essas estatísticas foram obtidas para RQM, volume e sobreposição. Em cada tabela das próximas subseções, serão apresentadas as médias aritméticas dessas quantidades estatísticas, calculadas levando em consideração todos os experimentos. Adicionalmente, as mesmas quantidades estatísticas serão apresentadas em relação à cobertura do conjunto de biclusters nos experimentos efetuados.

Note que essas estatísticas são calculadas em cima do valor médio do conjunto de biclusters, ou seja, no caso do RQM será apresentada a média de seu valor médio no conjunto de biclusters nas 30 repetições. O mesmo se aplica à média do volume e da sobreposição.

De forma a facilitar a visualização, diagramas de caixa serão apresentados para verificar possíveis sobreposições dos valores médios obtidos para cada algoritmo, diante da variação dos mesmos. Os diagramas de caixa permitem comparar se todas as quantidades estatísticas calculadas para um algoritmo que obteve melhor média são melhores do que aquelas associadas aos outros algoritmos para todos os casos, levando em conta sua variação e valores máximos e mínimos.

Para assegurar a validade das conclusões feitas com base nos resultados, foi utilizado o teste de hipótese de análise de variância não-paramétrica Kruskal-Wallis (Gibbons & Chakraborti, 2003), com o critério de Tukey-Kramer (Hochberg & Tamhane, 1987). Esse teste estatístico ajuda a determinar se a mediana da distribuição de vários conjuntos diferentes de amostras contém diferenças significativas. Uma vantagem desse método em relação a outros métodos de teste de hipótese é que ele não assume que os conjuntos de amostras apresentam uma distribuição normal, mas apenas que esses conjuntos apresentam uma mesma distribuição qualquer. Outra vantagem é que ela é capaz de lidar com vários grupos de uma única vez (o teste-t, por exemplo, só consegue comparar dois grupos por vez). É necessário notar, porém, que como a qualidade do conjunto de biclusters envolve múltiplos objetivos, os resultados de significância estatística devem ser analisados globalmente. Para alguns

casos, histogramas comparativos entre distribuições serão plotados de forma a permitir uma análise mais completa.

Finalmente, gráficos mostrando a relação entre os valores médios de RQM, volume e cobertura de todos os algoritmos serão mostrados, de forma a definir uma superfície de *custo-benefício* que mostrará mais claramente os prós e contras de cada algoritmo em relação a cada base de dados.

4.7.1 Base de Dados *Yeast*

Iniciando pelos resultados anotados, a Tab. 4.1 mostra um comparativo entre os algoritmos de acordo com os valores de RQM. Inicialmente, é importante ressaltar que o objetivo principal desses algoritmos é manter tal valor abaixo da restrição estipulada por δ, o que todos os algoritmos, exceto os métodos *BIC-aiNet* e *MOM-aiNet*, cumprem. Analisando os valores médios, é possível notar que a *MOM-aiNet* mantém biclusters muito próximos do limiar, ou seja, procuram biclusters próximos de δ que maximizem o volume. Os menores valores médio e de mediana foram obtidos pelo método *Cheng & Church*, mas com o custo de redução do valor médio de volume conforme pode ser visto na Tab. 4.2. O segundo menor RQM foi obtido pelo algoritmo *bicACO*, porém sem ocasionar a redução do volume médio dos biclusters obtidos. As 2 versões do algoritmo SwarmBcluster obtiveram o terceiro menor RQM, notando que quanto mais apurada a busca, maior seu valor de RQM, porém com um volume médio maior.

Tab. 4.1: Média das quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo do RQM, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados Yeast.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	$147,90 \pm 31,50$	$146,49$	$6,30$	$262,58$
SwarmBcluster (15)	$166,45 \pm 39,27$	$165,57$	$1,12$	$285,33$
Cheng & Church	$62,44 \pm 82,82$	$27,91$	$0,00$	$289,91$
BIC-aiNet	$206,44 \pm 27,07$	$201,82$	$142,84$	$346,70$
MOM-aiNet	$292,43 \pm 35,35$	$298,51$	$3,23$	$329,51$
MOEABIC	$293,26 \pm 10,61$	$297,95$	$229,03$	$299,91$
bicACO	$138,98 \pm 8,75$	$140,58$	$110,98$	$168,90$

No gráfico apresentado na Fig. 4.4, é possível perceber que a maioria dos algoritmos se mantém estável em relação ao valor médio de RQM, não existindo sobreposição aparente na distribuição dos valores dentro dos experimentos. O método *Cheng & Church* apresentou uma variância maior, possivelmente por introduzir ruído aleatório incremental na base de dados. O método *BIC-aiNet* apresentou alguns *outliers*, mostrando também uma instabilidade de comportamento do algoritmo.

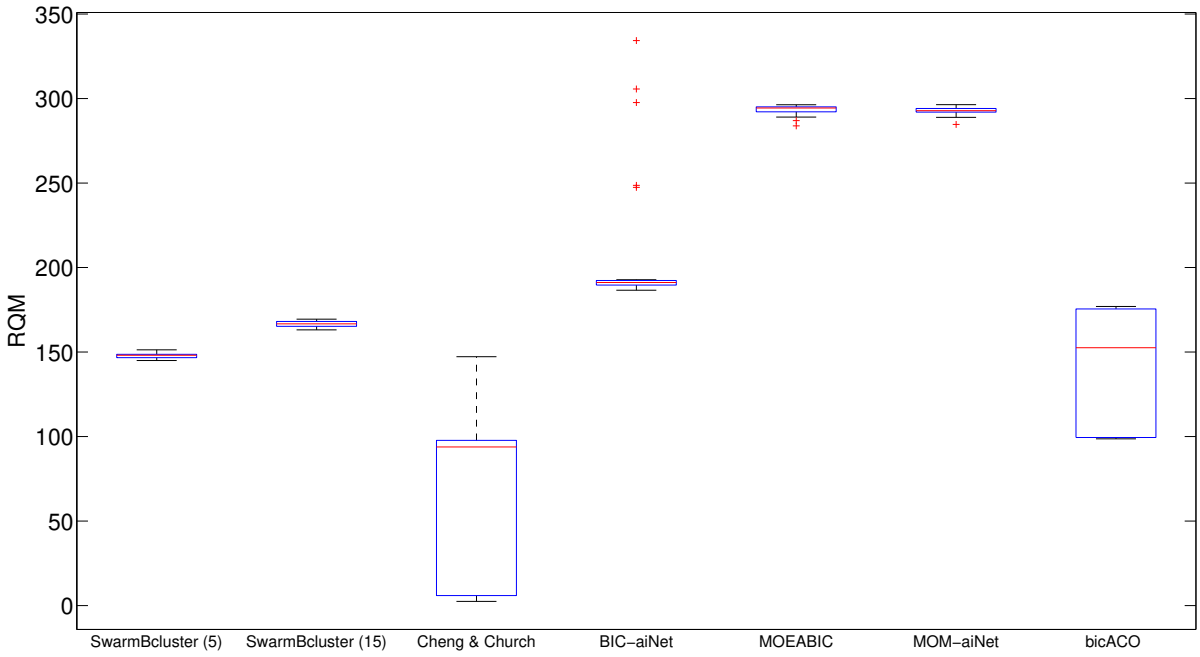


Fig. 4.4: Diagramas de caixa referentes ao RQM médio (considerando todos os biclusters produzidos em cada execução) de cada algoritmo para a base de dados *Yeast*.

Já o gráfico da Fig. 4.5 apenas confirma, através de testes estatísticos, o gráfico anterior. Além disso, esse gráfico mostra que a diferença entre o RQM dos métodos *SwarmBcluster*, *Cheng & Church* e *bicACO* não pode ser atestada, podendo pertencer à mesma distribuição, dentro da margem de confiança de 5%. Com isso, entre esses métodos, não é possível declarar aquele que tem um melhor desempenho quanto ao RQM.

Conforme já mencionado, a Tab. 4.2 relata as quantidades estatísticas referentes ao volume médio do conjunto de biclusters obtidos por cada algoritmo. A primeira coisa a notar é o alto valor de desvio padrão em todos os algoritmos. Isso ocorre pois não existe uma normalização no tamanho do bicluster e este é dependente diretamente das características da base de dados e da região explorada pelo bicluster. Analisando os valores médios, percebe-se que o algoritmo *MOEABIC* obteve um conjunto de biclusters com maior volume em relação a todos os outros algoritmos, que mantiveram valores médios próximos entre si, com exceção dos algoritmos *Cheng & Church* e *MOM-aiNet*. Porém, ao analisar os valores máximos e mínimos, percebe-se que o algoritmo *SwarmBcluster* encontrou biclusters maiores que o máximo encontrado pelas outras técnicas. Encontrou também valores muito menores, o que reduziu o seu valor médio. Essa diferença é atenuada no cálculo da mediana. Um dos fatores que contribui para esse resultado é o maior controle de sobreposição efetuado no *SwarmB-*

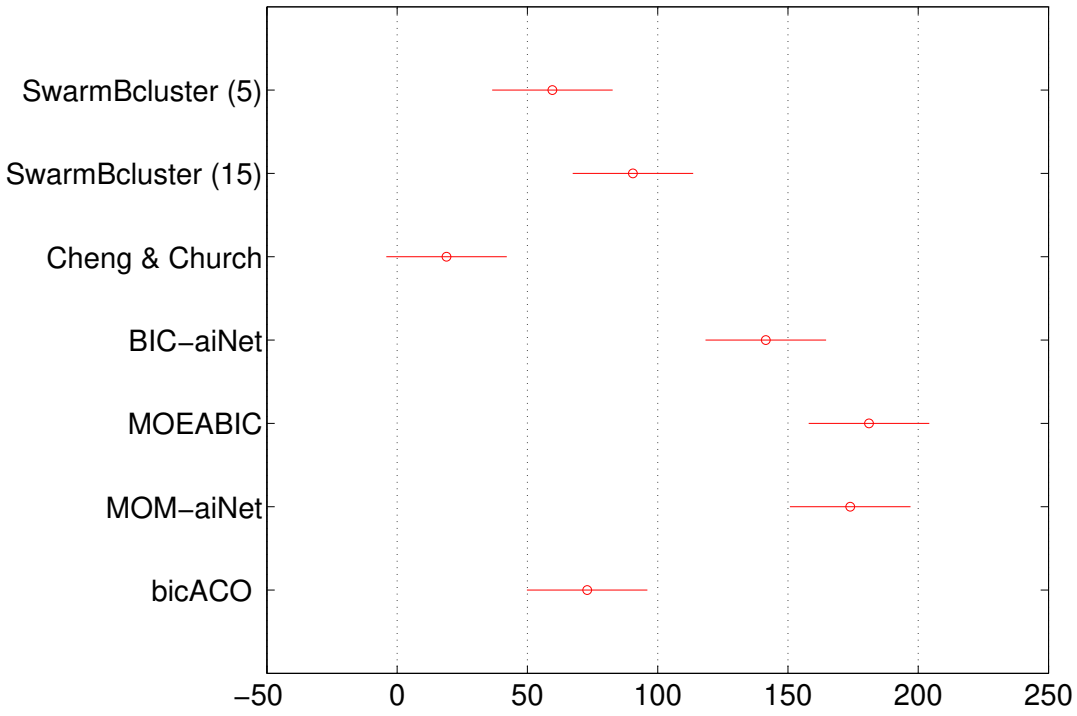


Fig. 4.5: Gráfico de teste de comparação múltipla de hipótese entre os algoritmos, conforme metodologia descrita anteriormente, referente ao RQM para a base de dados *Yeast*. Os valores do eixo x não tem significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

cluster que, após encontrar um número considerável de biclusters, busca por biclusters menores em regiões não exploradas, visando maximizar a cobertura.

Tab. 4.2: Média das quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo do volume, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados Yeast.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	3.169, 65 ± 1.711, 75	3.320, 65	16, 77	8.121, 57
SwarmBcluster (15)	3.897, 35 ± 2.241, 34	4.261, 21	16, 63	9.736, 23
Cheng & Church	59, 94 ± 317, 06	4, 12	2, 00	5.149, 10
BIC-aiNet	3.300, 01 ± 740, 71	3.439, 48	1.280, 77	4.739, 80
MOM-aiNet	1.812, 55 ± 680, 77	1.765, 48	27, 47	3.640, 73
MOEABIC	4.760, 36 ± 1.149, 23	4.568, 03	2.271, 70	7.272, 73
bicACO	3.158, 63 ± 568, 19	3.122, 37	2.085, 53	4.710, 27

Novamente, através dos diagramas de caixa, apresentada na Fig. 4.6, é possível perceber a alta variância do algoritmo *bicACO*, enquanto os outros métodos se apresentam estáveis quanto à repetição dos experimentos. Igualmente ao caso anterior, o algoritmo *BIC-aiNet* apresentou *outliers*, compatíveis com o gráfico anterior, em que representavam maior valor de RQM, o que permite um maior volume.

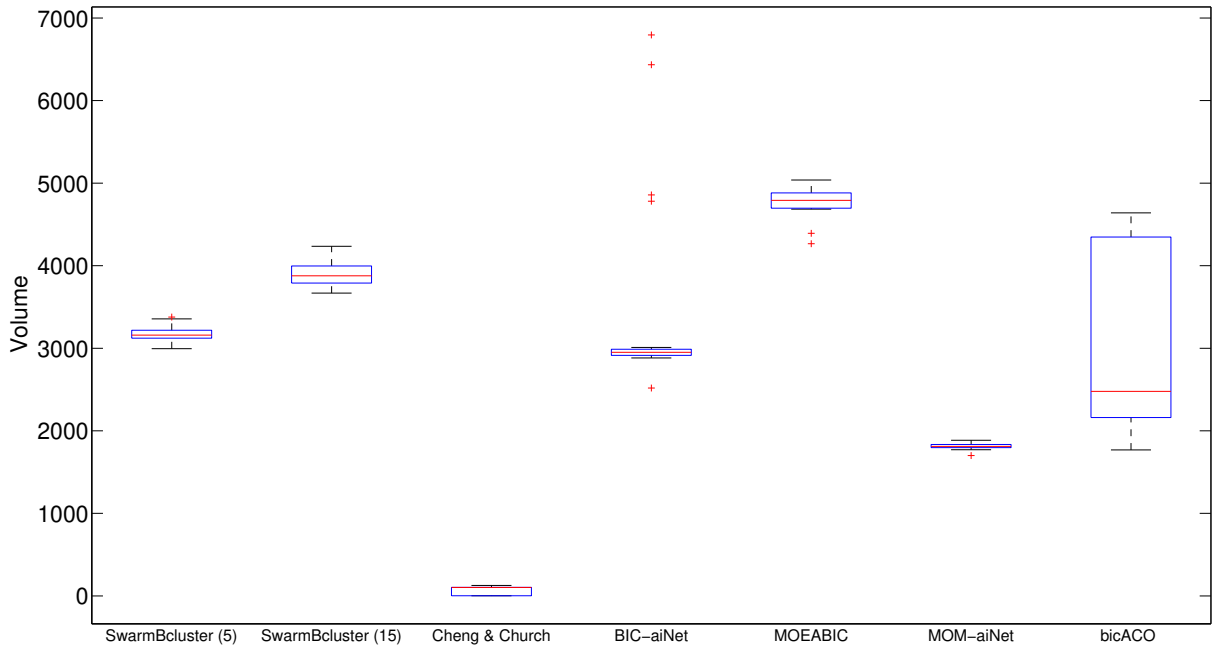


Fig. 4.6: Diagramas de caixa referentes ao volume médio de cada algoritmo para a base de dados *Yeast*.

Os testes estatísticos referentes ao volume, mostrados na Fig. 4.7, indicam que a distribuição dos métodos *SwarmBcluster*, *BIC-aiNet*, *MOEABIC* e *bicACO* apresentam um comportamento similar, o que significa que não se pode afirmar qual, dentre esses, possui o melhor volume médio. Esse tipo de resultado já era esperado através da Tab. 4.2: a média obtida pelo *MOEABIC* é significativamente maior que aquela obtida pelo *SwarmBcluster*, mas a média do maior valor de volume do *SwarmBcluster*, por sua vez, é maior que a obtida pelo *MOEABIC*.

O maior controle em relação à sobreposição dos biclusters pode ser percebido na Tab. 4.3, onde os valores médios e medianos do SwarmBcluster são menores que aqueles obtidos pelas outras técnicas que apresentam volume comparável ou maior. Apesar disso, todos os algoritmos mantiveram um valor mediano baixo, o que implica baixa redundância. Apenas o algoritmo *MOEABIC* obteve um valor máximo próximo de 1, que indica redundância total entre dois biclusters. Os algoritmos *Cheng*

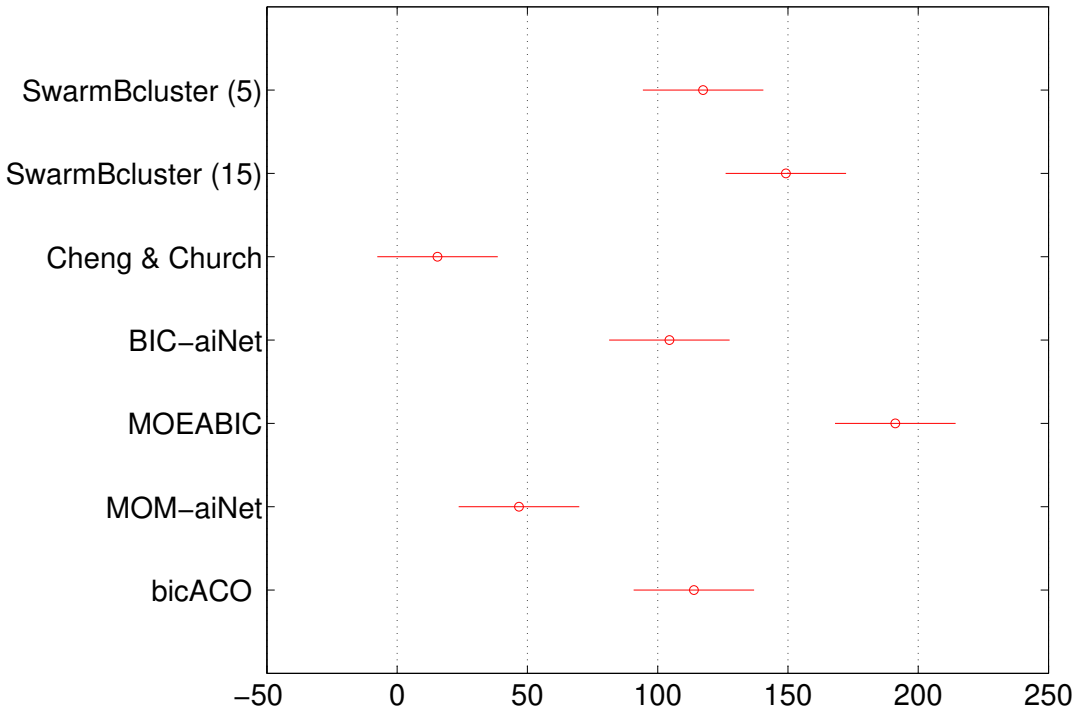


Fig. 4.7: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente ao volume para a base de dados *Yeast*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

& Church e *MOM-aiNet* obtiveram valores menores de sobreposição, porém os volumes médios são proporcionalmente menores também.

Tab. 4.3: Média das quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo da sobreposição, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados Yeast.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	$0,10 \pm 0,09$	$0,10$	$0,00$	$0,65$
SwarmBcluster (15)	$0,12 \pm 0,11$	$0,11$	$0,00$	$0,69$
Cheng & Church	$0,00 \pm 0,01$	$0,00$	$0,00$	$0,58$
BIC-aiNet	$0,35 \pm 0,15$	$0,38$	$0,02$	$0,81$
MOM-aiNet	$0,03 \pm 0,02$	$0,03$	$0,00$	$0,16$
MOEABIC	$0,19 \pm 0,11$	$0,17$	$0,00$	$0,96$
bicACO	$0,23 \pm 0,13$	$0,23$	$0,00$	$0,74$

Nos diagramas de caixa da Fig. 4.8, nota-se uma variância um pouco maior em todos os algoritmos do que a obtida em outros critérios. O algoritmo *bicACO* continua obtendo uma alta variância em seus valores. O método *BIC-aiNet* contém um único *outlier* destoando de sua média. Já os algoritmos *Cheng & Church* e *MOM-aiNet* se mantiveram mais estáveis, embora esse último tenha apresentado alguns *outliers*, porém não muito distantes de sua distribuição.

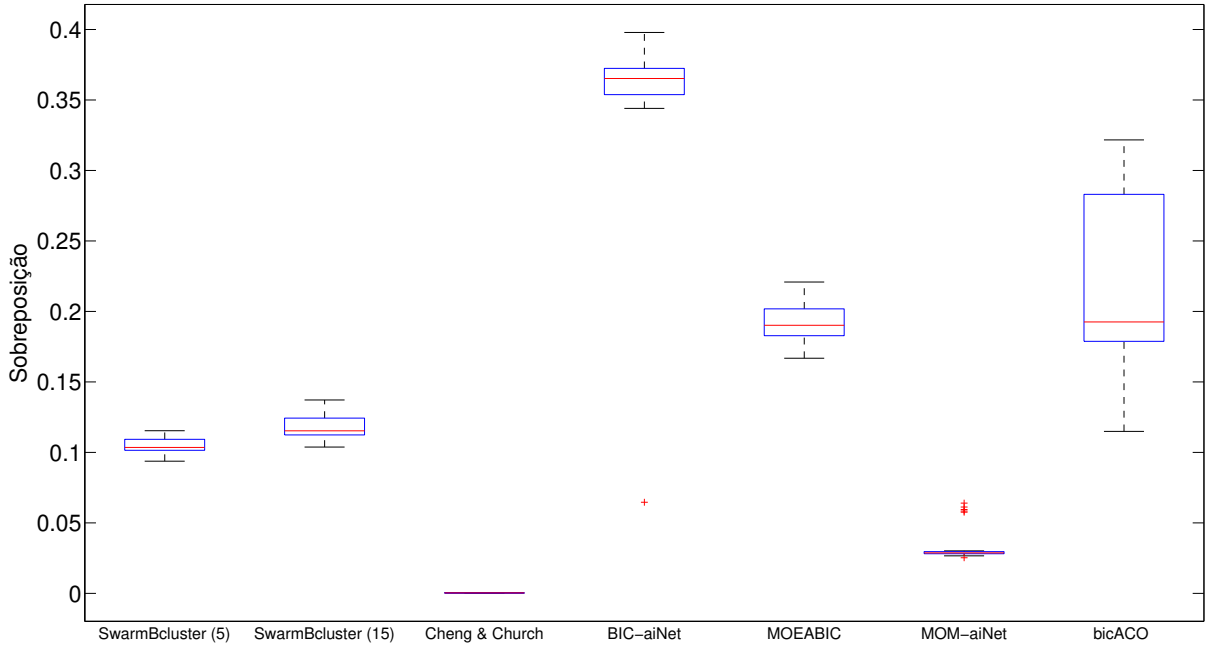


Fig. 4.8: Diagramas de caixa referentes à sobreposição média de cada algoritmo para a base de dados *Yeast*.

Na Fig. 4.9, referente aos testes estatísticos da sobreposição, é possível afirmar que as duas versões do *SwarmBcluster* têm uma distribuição equivalente. Além disso, a versão com 5 formigas apresenta uma distribuição próxima à *MOM-aiNet*, enquanto a versão com 15 formigas apresentou uma distribuição próxima do algoritmo *MOEABIC*. Note que, nessa última comparação, a sobreposição das distribuições é menor e quase inexistente. Esse resultado mostra também o efeito do aumento do volume médio na sobreposição dos biclusters: ao aumentar o primeiro parâmetro, o segundo também tende a aumentar.

A Tab. 4.4 mostra a cobertura de cada algoritmo em relação à base completa. Verificando pelo valor médio, as 2 versões do *SwarmBcluster* obtiveram uma cobertura acima de 90%, com valores mínimos de 91% e máximo de 95%. Dos outros métodos, apenas a *MOM-aiNet* obteve uma cobertura máxima de 91%, porém com uma cobertura média de 88%. Apesar de um valor de cobertura menor,

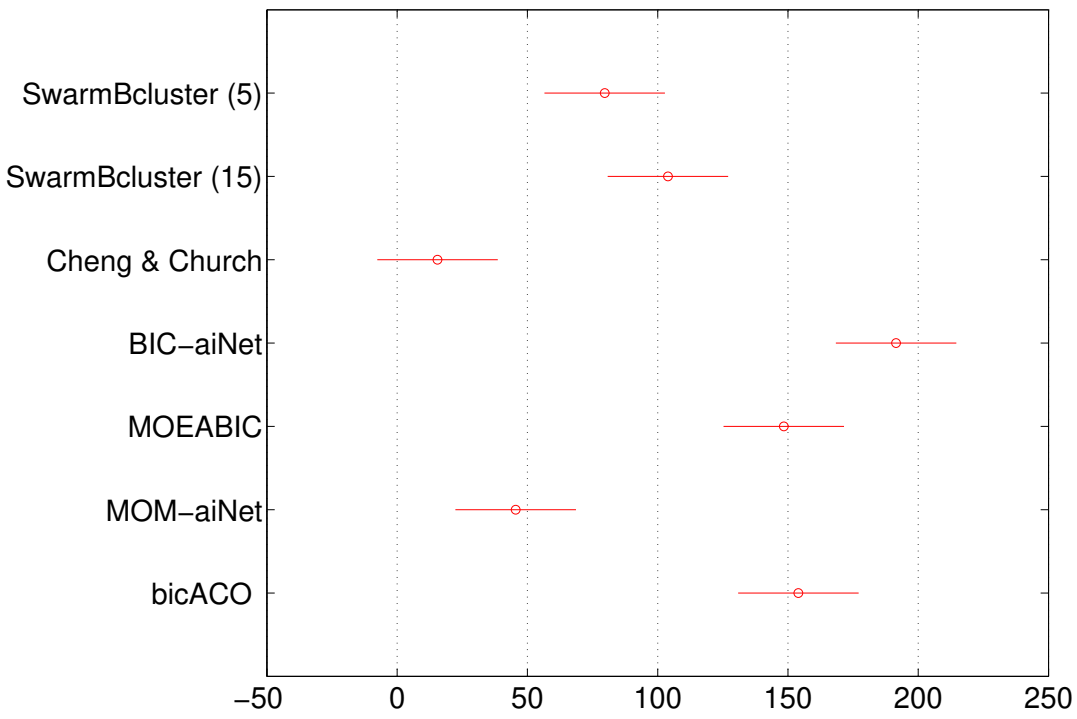


Fig. 4.9: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente à sobreposição para a base de dados *Yeast*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

nota-se que o volume da *MOM-aiNet* é consideravelmente menor também, indicando que ela consegue obter uma melhor diversidade dos biclusters. O *SwarmBcluster* consegue uma melhor exploração do espaço de busca, enquanto mantém a diversidade alta.

Os diagramas de caixa da Fig. 4.10 mostram uma maior estabilidade nas duas versões do *SwarmBcluster* em relação aos outros métodos. Os métodos *MOEABIC* e *MOM-aiNet* também apresentaram um comportamento estável, porém com maior variação dos valores. Os algoritmos *BIC-aiNet* e *bicACO* obtiveram baixa variância, porém diversos *outliers*. Finalmente, o algoritmo de *Cheng & Church* apresentou uma variação muito grande e, mesmo assim, não foi capaz de atingir os valores obtidos por *SwarmBcluster*.

No que se refere os testes estatísticos, as distribuições dos resultados do *SwarmBcluster* apresentam um alto grau de similaridade, enquanto apenas a versão com 5 formigas apresenta similaridade com a distribuição do algoritmo *MOM-aiNet*. Isso indica que, de todos os métodos utilizados para

Tab. 4.4: Quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo da cobertura, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados *Yeast*.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	$0,92 \pm 0,00$	$0,92$	$0,91$	$0,92$
SwarmBcluster (15)	$0,94 \pm 0,00$	$0,94$	$0,93$	$0,95$
Cheng & Church	$0,50 \pm 0,05$	$0,49$	$0,46$	$0,70$
BIC-aiNet	$0,46 \pm 0,11$	$0,42$	$0,38$	$0,85$
MOM-aiNet	$0,88 \pm 0,01$	$0,89$	$0,84$	$0,91$
MOEABIC	$0,83 \pm 0,02$	$0,82$	$0,79$	$0,86$
bicACO	$0,50 \pm 0,05$	$0,49$	$0,46$	$0,70$

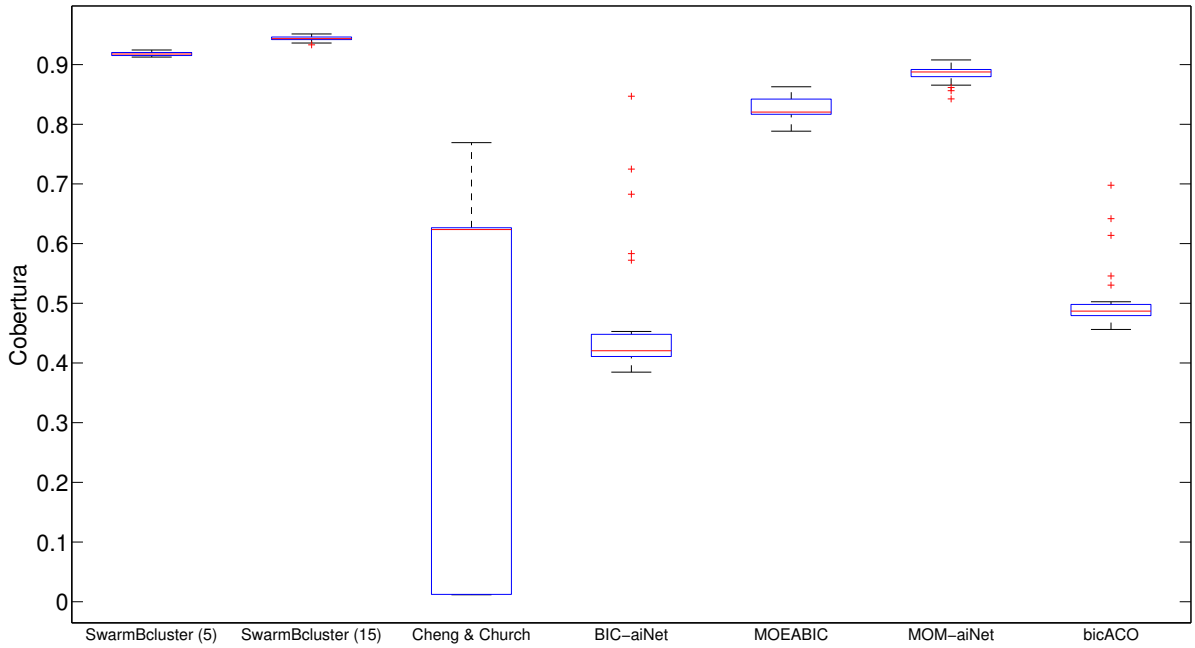


Fig. 4.10: Diagramas de caixa referentes à cobertura de cada algoritmo para a base de dados *Yeast*.

comparação, apenas a *MOM-aiNet* obteve resultados comparáveis com o *SwarmBcluster*.

Nesse ponto, percebe-se que não é possível definir um vencedor claro através da análise individual dos resultados, uma vez que o problema de biclusterização apresenta objetivos frequentemente conflitantes. Para facilitar a análise final, foram gerados quatro gráficos mostrando a relação *custo-benefício* entre os critérios analisados até então. Na Fig. 4.12(a), temos uma visão global em 3 dimensões dos resultados para todos os critérios. A região mais interessante desse gráfico gira em torno dos métodos

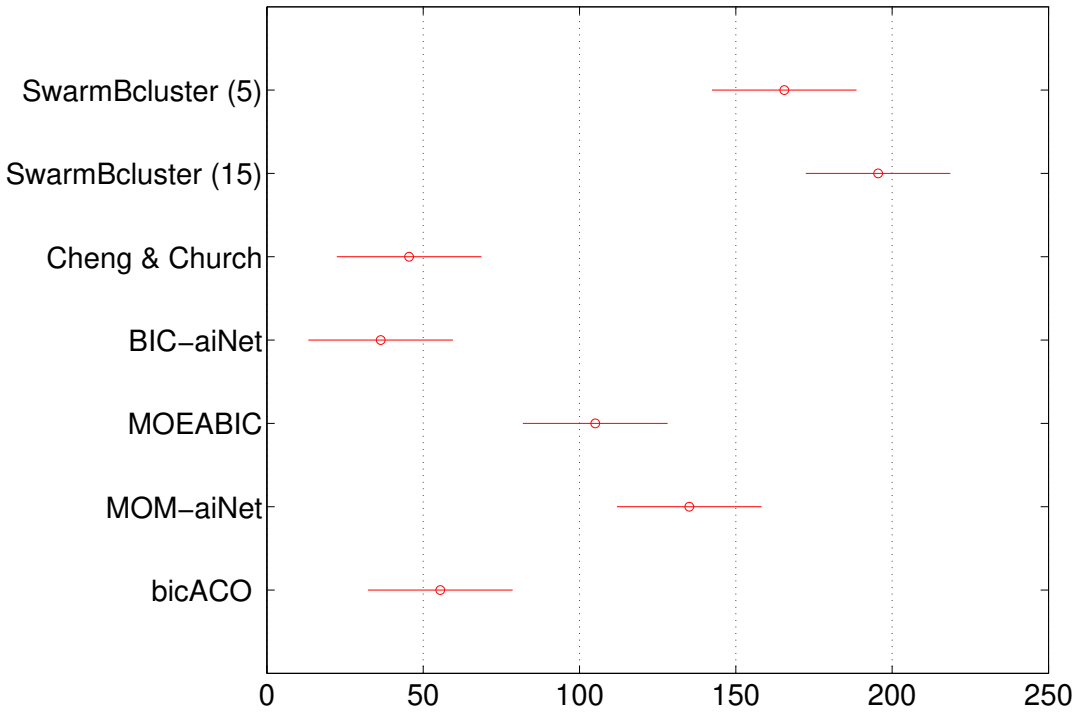


Fig. 4.11: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente à cobertura para a base de dados *Yeast*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

SwarmBcluster e *MOEABIC*, que conseguem maximizar a cobertura, mantendo os outros critérios equilibrados em relação aos outros métodos.

A Fig. 4.12(b) mostra que, apesar de um volume maior obtido pelo *MOEABIC*, os biclusters gerados por esse algoritmo mantêm um valor próximo do limite estipulado, enquanto a versão com 15 formigas do *SwarmBcluster* conseguiu minimizar o RQM mantendo um valor alto do seu volume médio. Na Fig. 4.12(c), temos a relação do RQM com a cobertura obtida pelos métodos. Aqui nota-se a clara vantagem do *SwarmBcluster*, uma vez que obtém a maior cobertura média mantendo o RQM com um valor baixo. Finalmente na Fig. 4.12(d) percebemos novamente uma disputa de objetivos entre o *SwarmBcluster* e o *MOEABIC*, enquanto o primeiro maximiza a cobertura, sem sacrificar tanto o volume, o segundo faz o oposto, maximiza o volume mas mantém uma taxa de cobertura ainda aceitável.

É importante notar que, dependendo da aplicação, podem ser desejáveis características diferentes do algoritmo a ser utilizado. Esses gráficos auxiliam na escolha daquele que melhor se adequa à

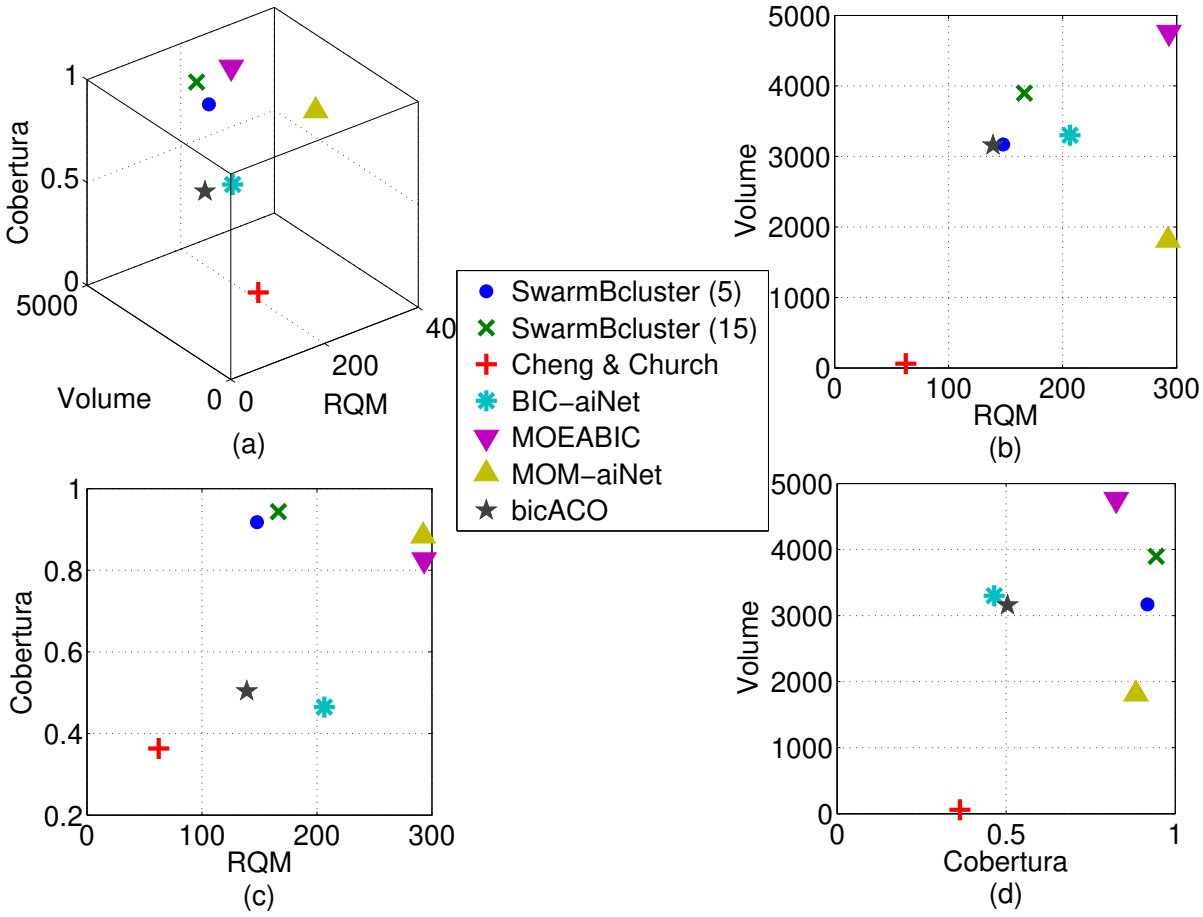


Fig. 4.12: Gráficos comparativos do custo-benefício de cada algoritmo para a base de dados *Yeast* referente a: (a) RQM, volume e cobertura; (b) RQM e volume; (c) RQM e cobertura e; (d) cobertura e volume.

situação desejada.

4.7.2 Base de Dados *Human*

Analisando os resultados obtidos para a base de dados *Human* através da Tab. 4.5, é possível perceber que o valor médio do RQM obtido pelo algoritmo *SwarmBcluster*, em suas duas versões, está abaixo dos outros métodos, com exceção do algoritmo *Cheng & Church* que, novamente, obteve o melhor desempenho nesse critério. Apesar disso, é possível perceber que a variação dos valores para o algoritmo *SwarmBcluster* é menor que a de *Cheng & Church*. Nas próximas tabelas, também será possível perceber que o método *Cheng & Church* obteve um desempenho inferior aos demais com relação aos outros critérios.

Tab. 4.5: Média das quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo do RQM, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados *Human*.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	922, 68 ± 67, 87	920, 42	726, 64	1.136, 58
SwarmBcluster (15)	949, 71 ± 62, 94	945, 93	774, 90	1.151, 80
Cheng & Church	742, 08 ± 145, 16	756, 27	319, 45	1.096, 53
BIC-aiNet	1.148, 85 ± 66, 25	1.146, 11	975, 84	1.341, 05
MOM-aiNet	1.047, 47 ± 377, 77	1.194, 10	0, 00	1.199, 95
MOEABIC	1.042, 22 ± 29, 13	1.043, 68	953, 92	1.102, 16

A Fig. 4.13 mostra que esses resultados obtidos mantêm uma variação aceitável e com poucas sobreposições dos valores obtidos pelas duas versões do *SwarmBcluster*, em relação aos outros métodos. A única ressalva fica por conta do algoritmo *Cheng & Church*, que possui alguns *outliers* com valores próximos aos obtidos por esses.

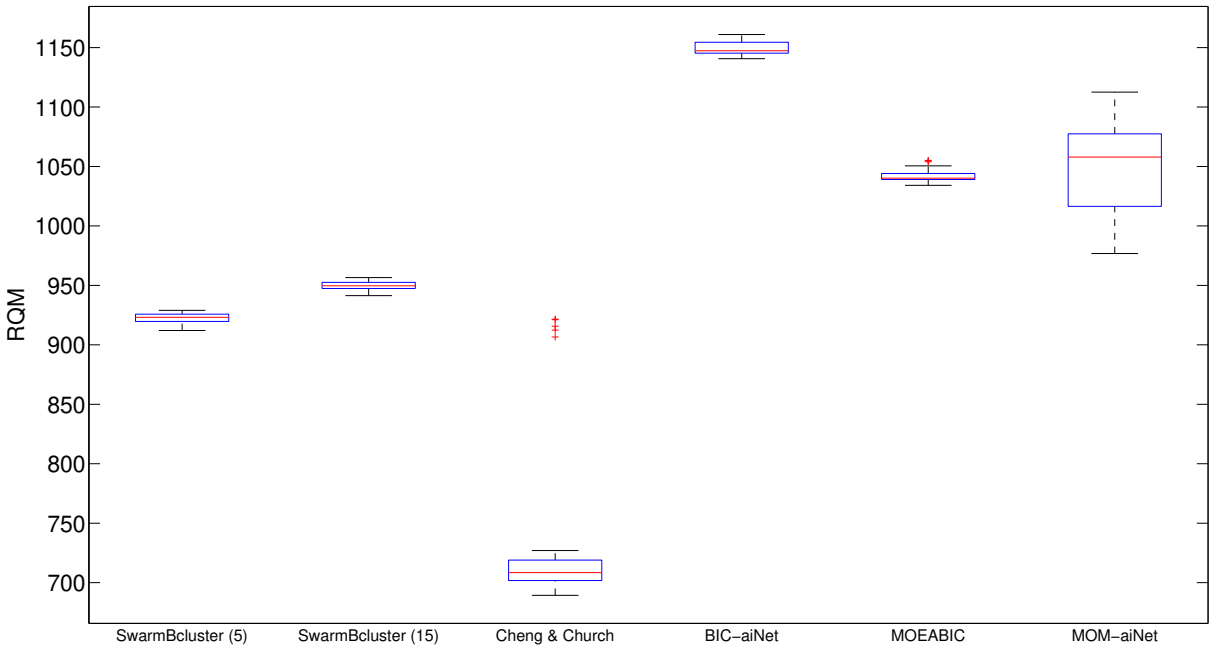


Fig. 4.13: Diagramas de caixa referentes ao RQM médio de cada algoritmo para a base de dados *Human*.

Nos testes estatísticos apresentados na Fig. 4.14, repete-se explicitamente o comportamento já apontado nos diagramas de caixa, onde as duas versões de *SwarmBcluster* possuem distribuições

equivalentes, quanto ao RQM, e distintas dos demais, com exceção do *Cheng & Church*, por conta dos *outliers* encontrados. Os métodos *MOEABIC* e *MOM-aiNet* apresentam distribuições muito similares entre si, levando à conclusão de que os resultados desses dois algoritmos podem ser considerados equivalentes. Já o algoritmo *BIC-aiNet* apresenta sua distribuição completamente diferente de todas as outras, mas apenas confirmando que ele obteve o pior resultado nessa avaliação.

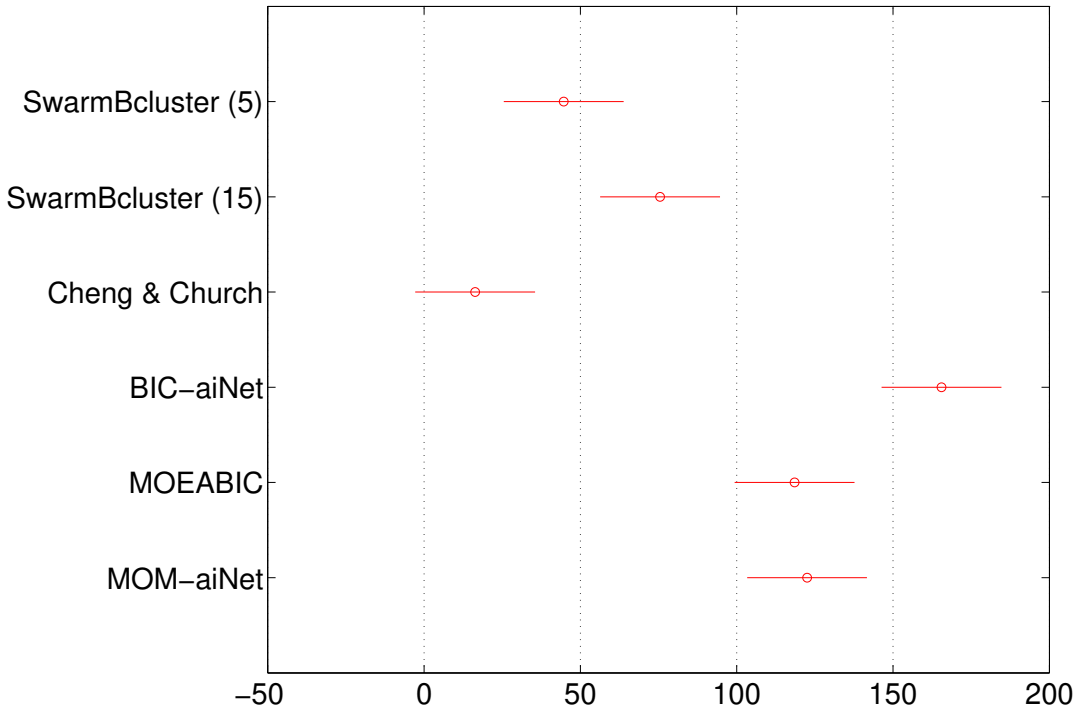


Fig. 4.14: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente ao RQM para a base de dados *Human*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

A Tab. 4.6 mostra os valores obtidos para o volume por cada um dos algoritmos. Nessa tabela, verifica-se que tanto a média quanto a mediana das duas versões do *SwarmBcluster* estão acima dos valores obtidos pelas outras abordagens, sendo que aquele que mais se aproxima é o algoritmo *BIC-aiNet*. Apesar disso, o desvio-padrão associado à média é maior no *SwarmBcluster* do que nos demais. As faixas de valores máximos e mínimos desse algoritmo também apresenta uma variabilidade maior que no caso dos outros métodos. Para exemplificar, o menor volume obtido pelo *SwarmBcluster* é muito menor que o menor obtido pelo *BIC-aiNet*. E o maior volume obtido pelo *SwarmBcluster* é consideravelmente maior que aquele obtido pelo *BIC-aiNet*. Mais à frente, será mostrado que essa

menor variação de volume da *BIC-aiNet* é um reflexo direto do aumento da sobreposição. É importante ressaltar também que o aumento do número de formigas e iterações permitiu um aumento significativo no volume médio obtido pelo *SwarmBcluster*.

Tab. 4.6: Média das quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo do volume, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados *Human*.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	15.412, 25 ± 4.089, 42	15.817, 77	1.336, 93	24.988, 87
SwarmBcluster (15)	17.666, 42 ± 4.164, 11	18.235, 70	2.328, 17	26.554, 60
Cheng & Church	478, 16 ± 818, 69	253, 15	29, 53	6.876, 40
BIC-aiNet	12.071, 38 ± 1.373, 46	12.084, 67	8.775, 00	15.475, 50
MOM-aiNet	3.156, 74 ± 1.710, 25	3.344, 12	111, 40	7.077, 63
MOEABIC	4.973, 96 ± 637, 64	4.942, 80	3.399, 97	7.011, 47

Os diagramas de caixa referentes ao volume, descritos na Fig. 4.15, mostram que o valor médio do volume sofre pouca variação em todos os métodos e, além disso, o algoritmo *SwarmBcluster*, em suas duas versões, se distanciam em relação aos demais.

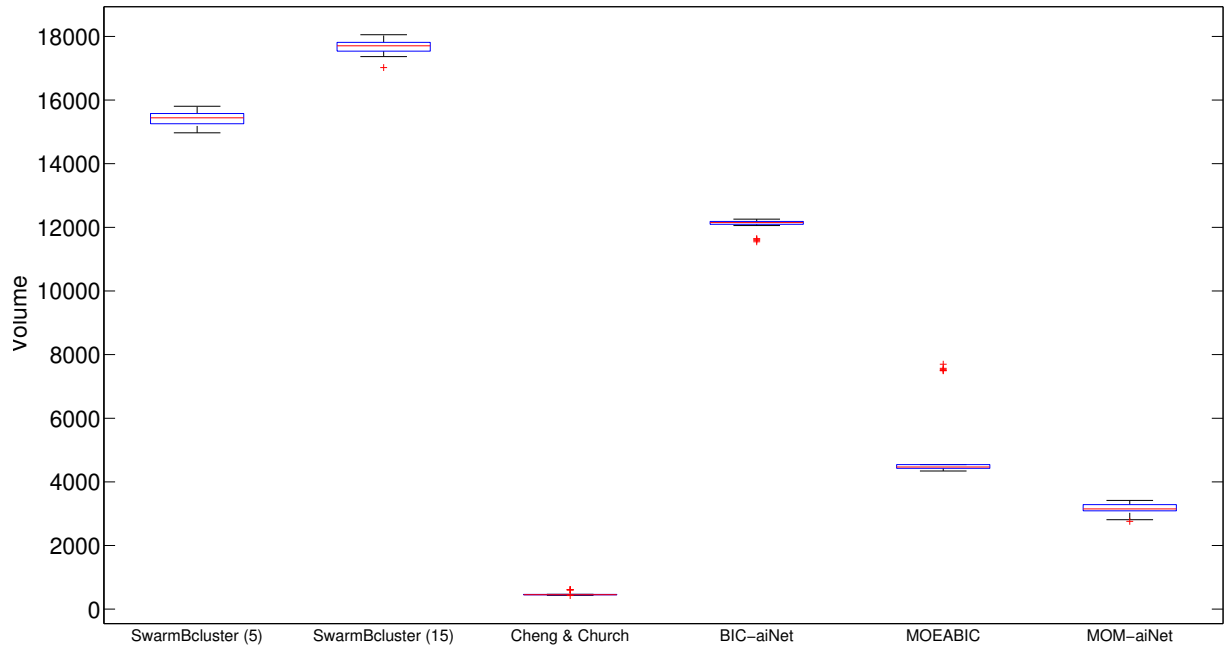


Fig. 4.15: Diagramas de caixa referentes ao volume médio de cada algoritmo para a base de dados *Human*.

Curiosamente, no gráfico referente aos testes estatísticos para o volume médio, apresentado na Fig. 4.16, extrai-se que a distribuição dos resultados da versão de *SwarmBcluster* com 5 formigas é similar à distribuição obtida pelo algoritmo *BIC-aiNet*. Para melhor entender essa situação, foram gerados dois histogramas representando a distribuição dos valores de volume médio para o *SwarmBcluster (5)* e para a *BIC-aiNet*, conforme mostra a Fig. 4.17.

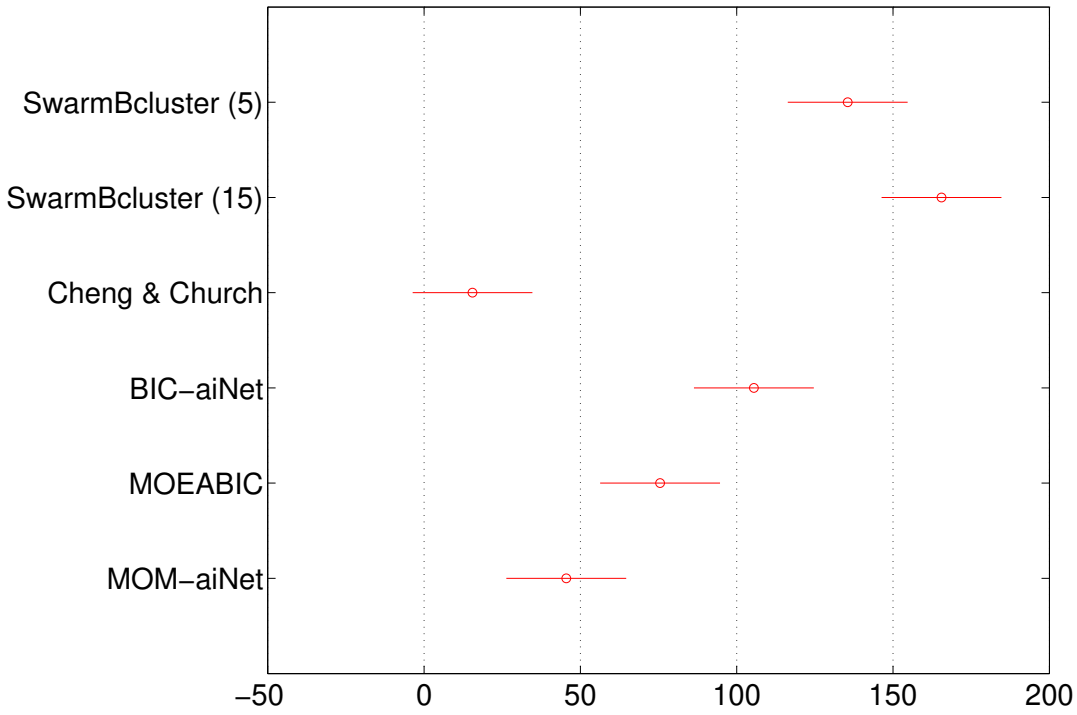
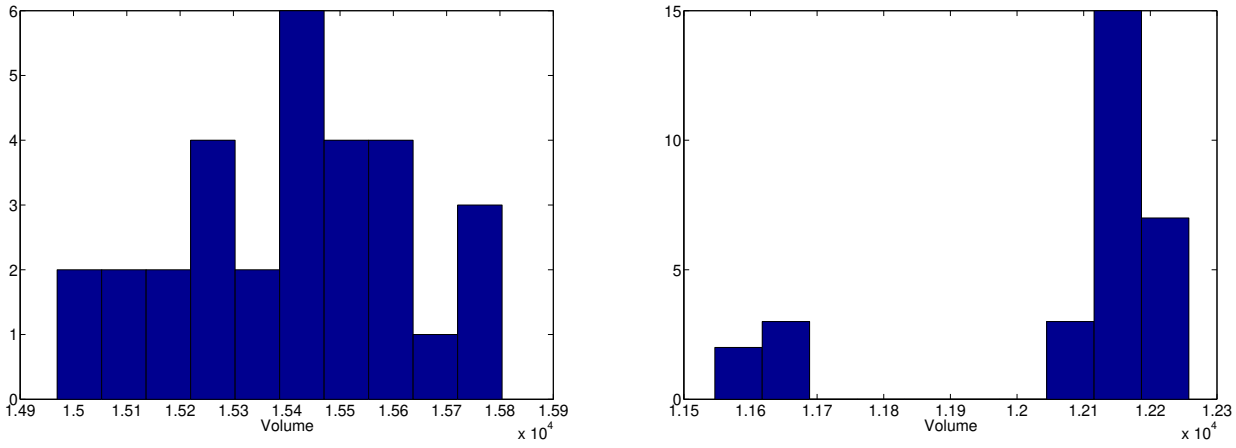


Fig. 4.16: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente ao volume para a base de dados *Human*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

A Fig. 4.17(a) mostra a distribuição obtida nos experimentos para o algoritmo *SwarmBcluster* com 5 formigas. Percebe-se que esse histograma forma aproximadamente uma distribuição centrada em torno de 15.400 com o desvio-padrão visualmente aproximado em 200. Já na Fig. 4.17(b) percebe-se a aproximação grosseira para uma distribuição centrada em torno de 12.200, mas com uma variância maior. Ao traçar a interseção entre as duas distribuições, existe a possibilidade de obter uma área em comum, mesmo que pequena, levando a crer que, com novos experimentos existe a chance de o algoritmo *BIC-aiNet* obter resultados máximos semelhantes aos resultados mínimos do *SwarmBcluster*.



(a) (b)

Fig. 4.17: Histograma da distribuição dos valores do volume médio nos experimentos efetuados na base de dados *Human* com os algoritmos: (a) *SwarmBcluster* e (b) *BIC-aiNet*.

Nesse experimento, a influência do volume médio dos biclusters no aumento da taxa de sobreposição pode ser percebida mais claramente conforme mostrado na Tab. 4.7. Os algoritmos que obtiveram os maiores volumes médios nessa base de dados foram aqueles com uma maior taxa de sobreposição. Vale notar, porém, que nenhum algoritmo obteve uma taxa de sobreposição completa, sendo que o valor máximo obtido, pelo algoritmo *MOEABIC*, foi de 88%. Além disso, apesar de ser o algoritmo com maior volume médio, o método *SwarmBcluster* manteve a taxa de sobreposição menor que o algoritmo *BIC-aiNet* e com valor máximo abaixo de 60%. Em contraste ao resultado do volume médio, embora a versão do *SwarmBcluster* com mais formigas e iterações tenha obtido um maior volume, a taxa de sobreposição não aumentou significativamente, ou seja, não foram incluídas redundâncias significativas com o aumento do volume.

Tab. 4.7: Média das quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo da sobreposição, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados *Human*.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	$0,20 \pm 0,08$	$0,20$	$0,00$	$0,54$
SwarmBcluster (15)	$0,23 \pm 0,09$	$0,23$	$0,00$	$0,57$
Cheng & Church	$0,00 \pm 0,00$	$0,00$	$0,00$	$0,00$
BIC-aiNet	$0,32 \pm 0,06$	$0,32$	$0,13$	$0,60$
MOM-aiNet	$0,01 \pm 0,01$	$0,01$	$0,00$	$0,12$
MOEABIC	$0,16 \pm 0,06$	$0,15$	$0,04$	$0,88$

Os diagramas de caixa na Fig. 4.18 indicam uma diferença significativa nos valores médios dos algoritmos, sendo que apenas o método *MOEABIC* apresenta *outliers* que destoam consideravelmente de seu resultado médio. E, assim como na situação anterior, os testes estatísticos apontaram similaridades entre a distribuição dos resultados dos algoritmos *SwarmBcluster (5)* e *MOEABIC* e dos algoritmos *SwarmBcluster (15)* e *BIC-aiNet*, conforme a Fig. 4.19.

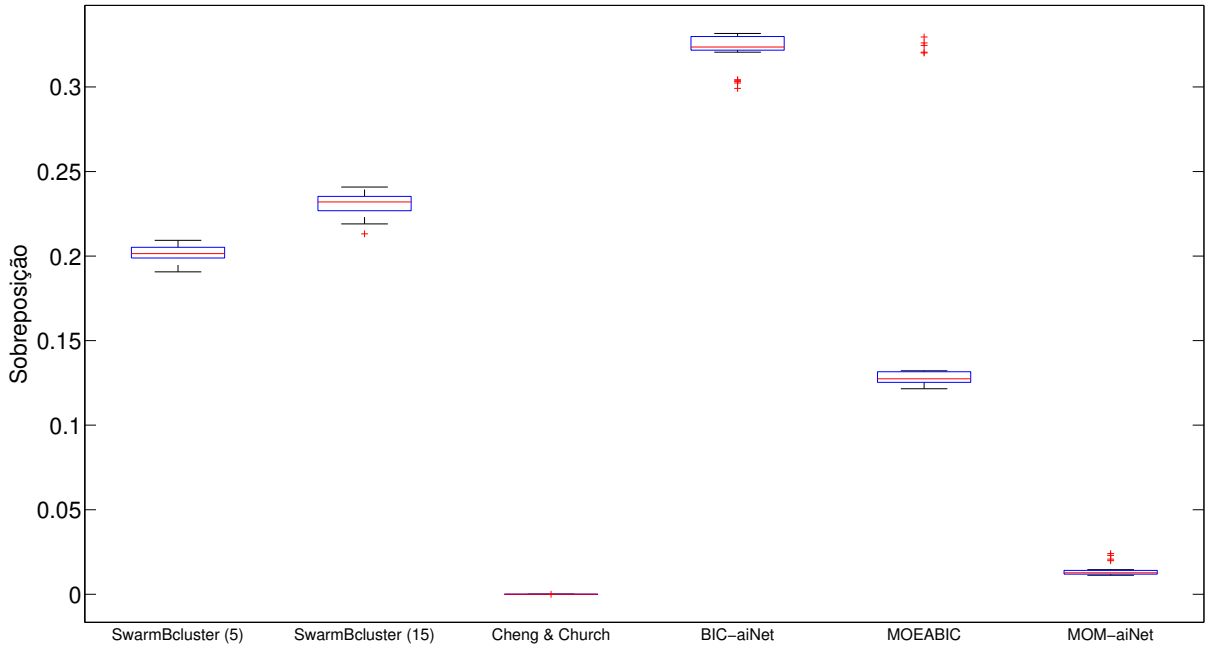


Fig. 4.18: Diagramas de caixa referentes à sobreposição média de cada algoritmo para a base de dados *Human*.

A Tab. 4.8 apresenta os resultados referentes à cobertura da base de dados para cada algoritmo. Nesses resultados, nota-se um percentual muito maior de cobertura com o algoritmo *SwarmBcluster* em relação aos outros métodos. O método que mais se aproximou dos resultados do *SwarmBcluster* foi novamente o algoritmo *MOM-aiNet*, que, curiosamente, foi o que obteve o segundo menor volume médio, mostrando novamente que esse método é eficiente na manutenção de diversidade. Porém, mesmo o valor máximo da *MOM-aiNet* é significativamente menor que o valor mínimo obtido pelo *SwarmBcluster*. Outro ponto que deve ser notado é que as duas versões do *SwarmBcluster* obtiveram uma taxa de cobertura similar, o que, juntando com os resultados dos outros critérios, permite concluir que o aumento do volume aumentou a ocorrência de redundância, porém limitando a taxa de sobreposição entre os elementos do conjunto de biclusters.

Os diagramas de caixa da Fig. 4.20 novamente mostram a disparidade entre os resultados obtidos

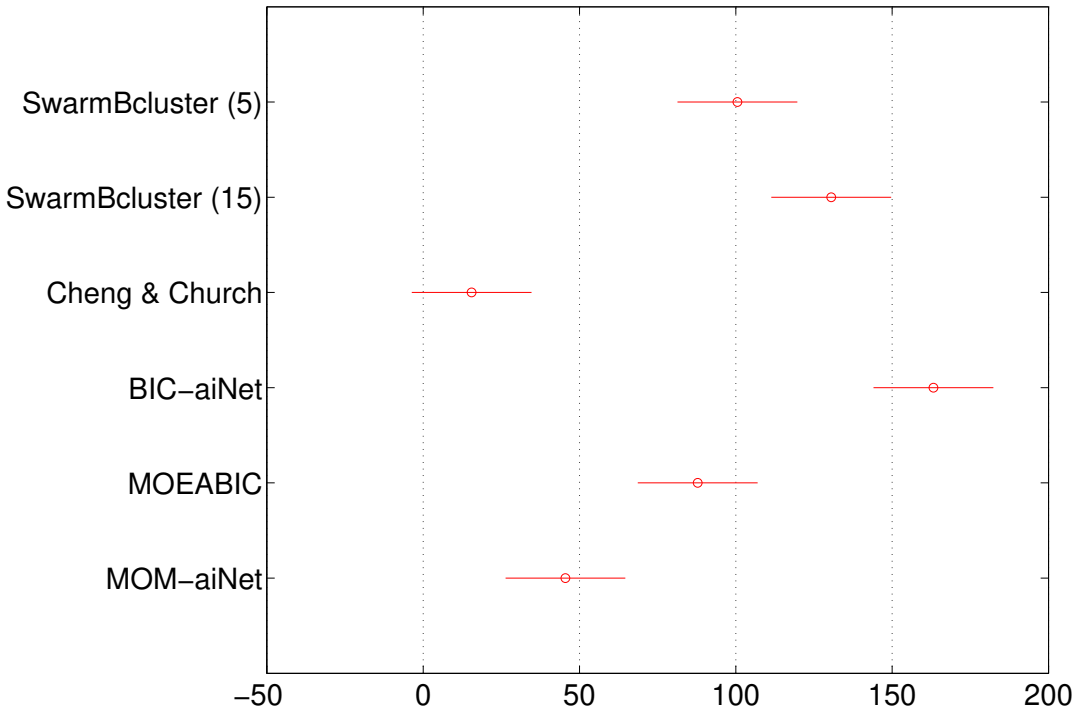


Fig. 4.19: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente à sobreposição para a base de dados *Human*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

Tab. 4.8: Quantidades estatísticas de média aritmética, desvio-padrão, mediana, valor mínimo e valor máximo da cobertura, obtidas pelos algoritmos em 30 repetições independentes dos experimentos para a base de dados *Human*.

Algoritmo	Média ± Desv. Pad.	Mediana	Mínimo	Máximo
SwarmBcluster (5)	$0,52 \pm 0,00$	$0,52$	$0,51$	$0,53$
SwarmBcluster (15)	$0,52 \pm 0,01$	$0,52$	$0,51$	$0,53$
Cheng & Church	$0,25 \pm 0,03$	$0,23$	$0,22$	$0,32$
BIC-aiNet	$0,23 \pm 0,00$	$0,23$	$0,22$	$0,23$
MOM-aiNet	$0,42 \pm 0,01$	$0,42$	$0,41$	$0,43$
MOEABIC	$0,22 \pm 0,02$	$0,23$	$0,17$	$0,23$

pelo *SwarmBcluster* e pelos outros métodos. Mostram também que, apesar de alguns *outliers*, todos os métodos se apresentam de forma estável. Já o teste estatístico representado na Fig. 4.21 mostra que

as distribuições obtidas pelos resultados do *SwarmBcluster* têm diferença significativa em relação aos outros métodos, apesar de se aproximarem da distribuição obtida pela *MOM-aiNet*.

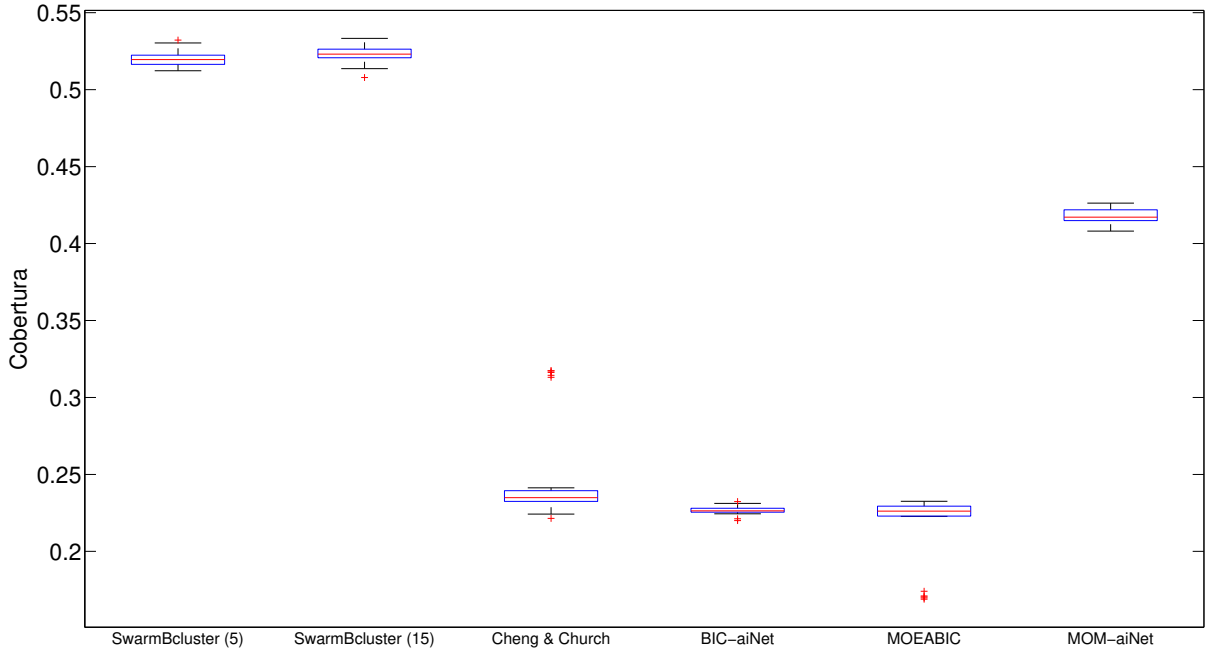


Fig. 4.20: Diagramas de caixa referentes à cobertura de cada algoritmo para a base de dados *Human*.

Finalmente, na Fig. 4.22 é possível observar o *custo-benefício* de cada algoritmo diante dos três critérios principais: RQM, volume e cobertura. Nessa figura, é possível perceber que o *SwarmBcluster* se mostra superior a todas as outras técnicas em relação ao custo-benefício, perdendo apenas para o critério de RQM para o algoritmo de *Cheng & Church*, embora este tenha os piores resultados em todos os outros critérios.

4.7.3 Tempo Computacional

Em relação ao tempo computacional consumido por cada uma das abordagens, o gráfico da Fig. 4.23 resume o tempo médio de cada algoritmo em cada uma das bases. Conforme é possível perceber, para a base de dados *Yeast*, Fig. 4.23(a), a maior diferença encontra-se nos algoritmos *Cheng & Church*, que tem o menor tempo por ser uma simples heurística construtiva; *bicACO*, que possui um tempo muito maior do que os outros métodos; e a versão que utiliza 15 formigas e 15 iterações do *SwarmBcluster*, que tem um custo computacional relativamente maior que a outra versão.

Na Fig. 4.23(b), referente à base de dados *Human*, a diferença entre o custo computacional de

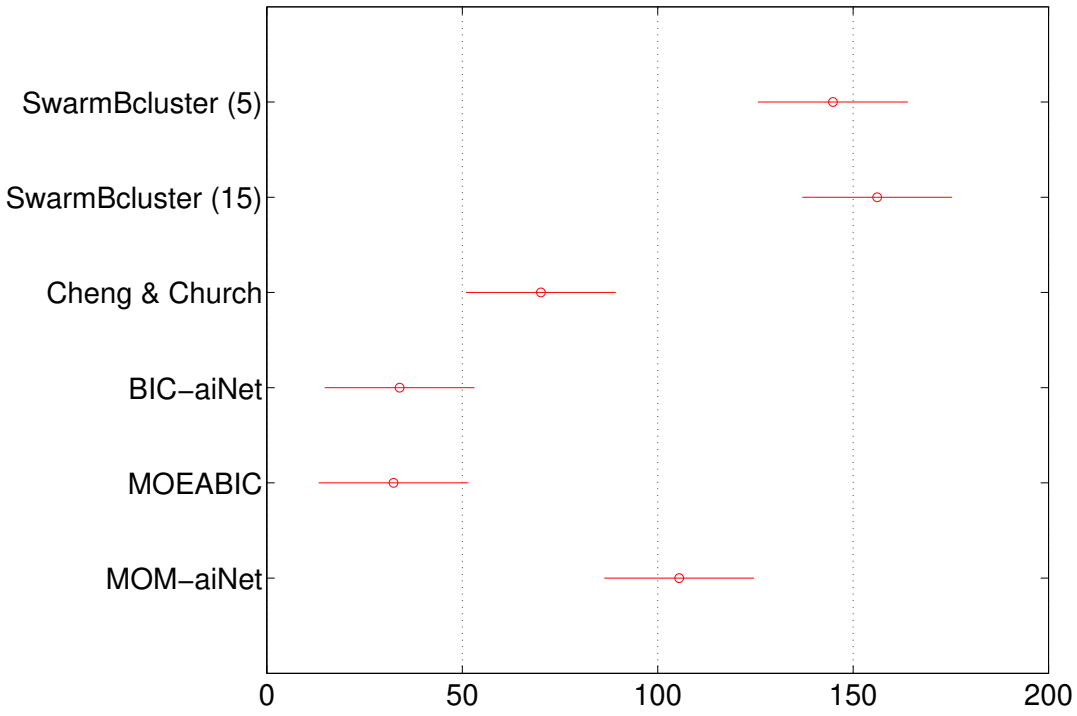


Fig. 4.21: Gráfico de teste de hipótese de comparação múltipla entre os algoritmos, conforme metodologia descrita anteriormente, referente à cobertura para a base de dados *Human*. Os valores do eixo x não têm significado prático, apenas denotam a proporção da diferença entre as médias dos algoritmos.

cada abordagem torna-se mais destacada. A versão com 15 formigas do *SwarmBcluster* novamente gerou um custo muito maior do que a versão com 5 formigas. Os métodos *MOEABIC* e *MOM-aiNet* tiveram um custo computacional maior que a *SwarmBcluster* com 5 formigas e o método *BIC-aiNet* e, novamente, por suas características, o *Cheng & Church* foi o algoritmo com menor custo.

Desconsiderando o custo computacional do *bicACO* e do *SwarmBcluster* com 15 formigas, e do *Cheng & Church*, por ser uma heurística simples, não se pode afirmar qual dos algoritmos realmente tem um melhor desempenho, pois diversos fatores interferem nesse resultado, como otimização de código, estado atual da máquina e plataforma. Porém, esses resultados mostram que o custo computacional desses algoritmos são equivalentes, o que leva à conclusão de que a escolha dentre esses algoritmos para um problema de biclusterização deve ser feita a partir da qualidade dos resultados. Outro ponto que deve ser levado em consideração é de que definir um tempo maior de otimização para o *SwarmBcluster* pode não ser interessante pois, apesar do aumento do volume médio, os outros cri-

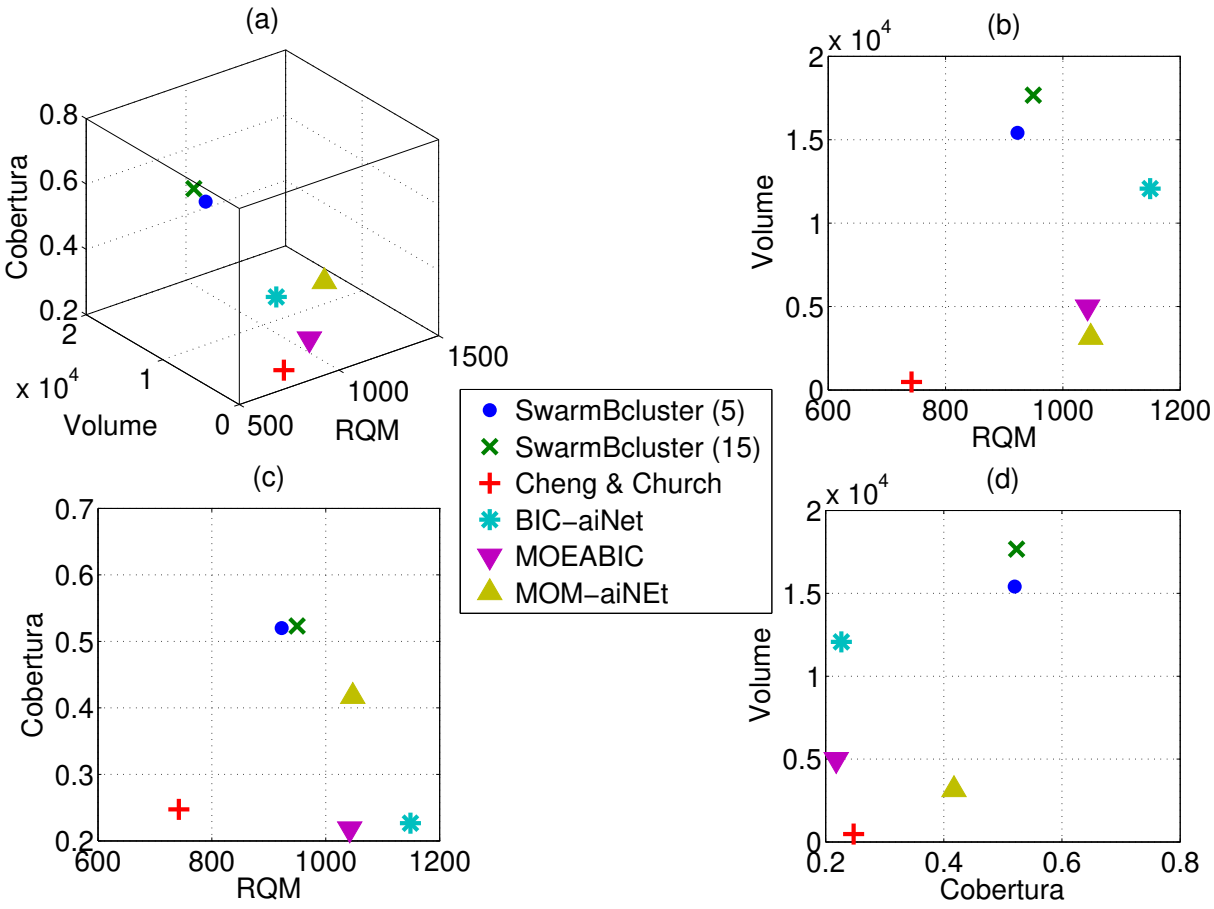
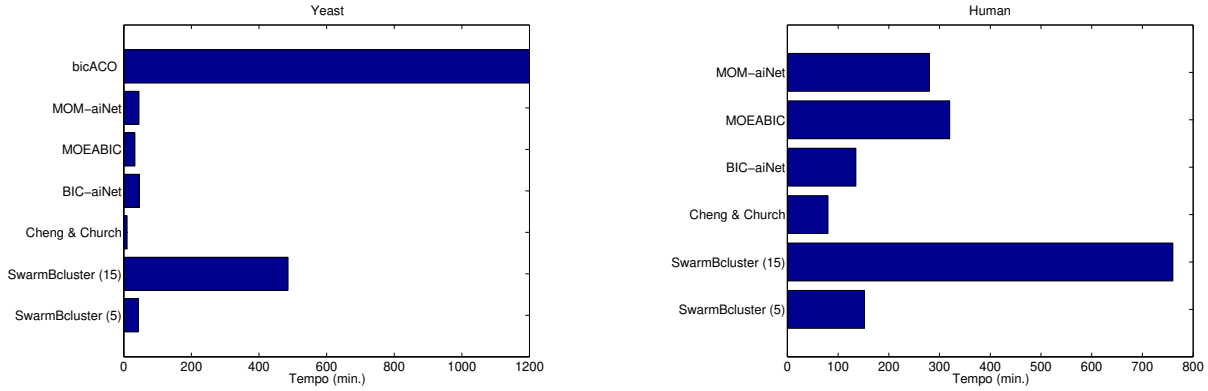


Fig. 4.22: Gráficos comparativos do custo-benefício de cada algoritmo para a base de dados *Human* referente a: (a) RQM, volume e cobertura; (b) RQM e volume; (c) RQM e cobertura e; (d) cobertura e volume.

térios não obtiveram melhora significativa, mas o custo computacional aumentou significativamente.

4.7.4 Teste Complementar 1: Capacidade de Eliminar Ruído

Conforme explicado no Cap. 3, as técnicas de biclusterização que buscam por biclusters *imperfeitos*, como biclusters coerentes, assumem que a região da base de dados na verdade contém um ruído de valor baixo – no caso estudado aqui, limitado pelo valor de δ. Com isso, as técnicas de biclusterização baseadas na coerência aditiva não só encontram biclusters com essas características mas também encontram um modelo, livre de ruído, para a região de cada bicluster. Vale notar que esse modelo é dito livre de ruído pois, ao calcular os elementos do bicluster partindo desse modelo, obtém-se um bicluster perfeitamente coerente, porém isso não significa que esses valores sejam equivalentes aos



(a) (b)

Fig. 4.23: Gráfico comparativo do tempo computacional médio, em minutos, de cada algoritmo para as bases de dados (a) *Yeast* e (b) *Human*.

valores da base de dados sem ruído.

Como um teste complementar, utilizando o método *SwarmBcluster* foram gerados 500 biclusters para a base de dados *ArtDatasetRuido* descrita no Cap. 2. A partir desses biclusters, foi utilizada a Eq. 3.19, reproduzida aqui na Eq. 4.15:

$$a_{ij} = a_{Ij} + a_{iJ} - a_{IJ}, \tag{4.15}$$

onde a_{ij} é o valor do elemento da linha i e coluna j, a_{Ij} é a média da coluna j do bicluster, a_{iJ} é a média da linha i do bicluster, e a_{IJ} é a média dos valores do bicluster.

Os valores de cada elemento do bicluster, definidos pelo seu modelo, foram recalculados utilizando a Eq. 4.15 e estes formaram uma nova base de dados, que foi comparada com a base *ArtDataset*, livre de ruído, calculando a diferença média absoluta entre seus valores.

Esse procedimento pode ser ilustrado a partir da Fig. 4.24. A Fig. 4.24(a) ilustra um bicluster extraído de uma base de dados ruidosa, enquanto a Fig. 4.24(b) representa esse mesmo bicluster com os valores provenientes da base sem ruído. Utilizando as médias do bicluster ruidoso em conjunto com a Eq. 4.15, recalcula-se os valores desse bicluster, resultando em um novo bicluster ilustrado na Fig. 4.24(c), com valores mais próximos da base de dados original.

Os parâmetros utilizados para este experimento foram definidos experimentalmente como 5 formigas e 5 iterações, $delta = 1,6$, $col\_min = 70$ e o restante dos parâmetros iguais aos experimentos anteriores. Na Tab. 4.9, verifica-se a redução de aproximadamente 33% do ruído. Vale notar que a utilização de técnicas especificamente voltadas para a redução de ruído possivelmente conseguiriam obter uma redução ainda maior. Porém não é esse o objetivo das técnicas de biclusterização, mas

$$\begin{bmatrix} 1,1 & 2,2 & 2,9 \\ 4,2 & 4,8 & 6,1 \\ 6,8 & 8,2 & 8,8 \\ 10,2 & 10,9 & 12,3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \quad \begin{bmatrix} 1,1 & 2,0 & 3,0 \\ 4,1 & 5,0 & 6,0 \\ 7,0 & 7,9 & 8,9 \\ 10,2 & 11,1 & 12,1 \end{bmatrix}$$

(a) (b) (c)

Fig. 4.24: Exemplo ilustrativo do procedimento realizado para recalcular os valores encontrados de uma base de dados ruidosa: (a) bicluster encontrado dentro de uma base de dados ruidosa; (b) o mesmo bicluster caso fosse extraído da base de dados sem ruído; e (c) novos valores obtidos utilizando o modelo definido pelo bicluster ruidoso.

apenas uma característica adicional de seus procedimentos.

Este experimento simples serviu apenas para mostrar que a técnica realmente tem essa capacidade, que servirá como base para um algoritmo de imputação de dados utilizando técnicas de biclusterização, o qual será descrito no próximo capítulo. Além disso, essa funcionalidade abre espaço para diversos outros estudos, como a utilização da biclusterização na redução de ruídos, o uso dela para melhorar técnicas tradicionais de eliminação de ruído, ou, até mesmo, o uso de técnicas de eliminação de ruído para melhorar a biclusterização.

Tab. 4.9: Erro absoluto entre a base ArtDatasetRuido e a base ArtDataset e entre a base formada pelo conjunto de biclusters gerados pelo algoritmo SwarmBcluster e a base ArtDataset.

ArtDatasetRuido × ArtDataset	**SwarmBcluster × ArtDataset**
0,9454	0,6335

4.7.5 Teste Complementar 2: Identificando Razões de Ausência de Dados

No Cap. 3, também foi mencionada a capacidade de as técnicas de biclusterização extraírem conhecimento de uma base de dados. Este conhecimento pode estar contido em apenas um subespaço definido por subconjuntos de objetos e atributos, o que dificulta a sua localização por conta dos valores restantes da base. Um possível conhecimento desejável para o problema de valores faltantes é o motivo, se existir, da ausência desses valores.

Conforme também visto no Cap. 2, esse motivo pode estar diretamente ligado aos valores de outros atributos (MAR), do próprio atributo (NMAR) ou ser independente de qualquer atributo (MCAR). Os dois primeiros motivos são particularmente interessantes para uma análise através da biclusterização, pois, no primeiro, essa técnica pode selecionar os atributos da base que contenham informação do motivo da ausência dos dados e, no segundo, pode ser possível encontrar uma tendência em outros

atributos de forma a encontrar objetos similares que possam indicar o motivo da ausência dos valores naquele determinado atributo.

Através das bases de dados geradas no Cap. 2, referentes à ausência de valores dos tipos MAR e NMAR, será mostrado agora que uma técnica de biclusterização, que visa uma adaptação da heurística construtiva proposta em conjunto com o algoritmo *SwarmBcluster*, é capaz de indicar os motivos da ausência de dados, conforme as regras estipuladas no Cap. 2.

Base de Dados MAR

Para obter informação sobre os dados faltantes do tipo MAR, uma heurística baseada naquela apresentada no Alg. 6 foi criada de forma a garantir a obtenção de informação na região dos dados faltantes. O procedimento inicia verificando quais atributos contêm dados faltantes. Para cada um desses atributos, é efetuado um procedimento similar à heurística construtiva proposta: gera-se o bicluster inicial composto pelas linhas que contêm valores faltantes naquele atributo e, iterativamente, escolhe-se um dos atributos como coluna inicial. A partir desse ponto, as outras colunas são inseridas sequencialmente e, para cada inserção, é verificado o número de linhas com RQM maior que um dado δ. A coluna inserida que possuir o menor número de linhas a serem removidas é então escolhida para fazer parte do bicluster, retirando as respectivas linhas. O processo continua até que um número mínimo de linhas seja atingido. Esse procedimento irá gerar $m-1$ biclusters, com m sendo o número de atributos, e o maior desses biclusters será utilizado para a análise.

A análise, para esse caso, será feita verificando quantos atributos existem no bicluster e quantos deles fazem parte dos atributos que definem os grupos que ocasionam os dados faltantes. O que se busca neste experimento é saber se um bicluster é capaz de identificar os atributos, e os valores atribuídos a esses, responsáveis pela ausência de dados.

Os resultados estão descritos na Tab. 4.10, na qual para cada atributo que contém valores faltantes (primeira coluna), são reportados o número de atributos que definem o motivo da ausência de valor (segunda coluna); o número de atributos no bicluster resultante (terceira coluna); a interseção entre esses dois conjuntos de atributos (quarta coluna); e a *taxa de descobrimento* (quinta coluna), calculada pela cardinalidade da interseção dos conjuntos de atributos dividida pelo número de atributos no bicluster. Essa taxa mostra o quanto do bicluster é útil para se definir o grupo a que pertencem os elementos com valores faltantes.

Para exemplificar a análise dos resultados, suponha que os valores do conjunto $\{2,3,4,5,6\}$ de atributos determinam as chances do valor do atributo 1 estar ausente, e o bicluster encontrado possui o conjunto de atributos $\{2,4,5,7,10\}$. A interseção desses conjuntos de atributos é $\{2,4,5\}$ e com a taxa de descobrimento calculada pode-se aferir que 60% dos atributos contidos no bicluster são determinantes para definir o motivo da ausência.

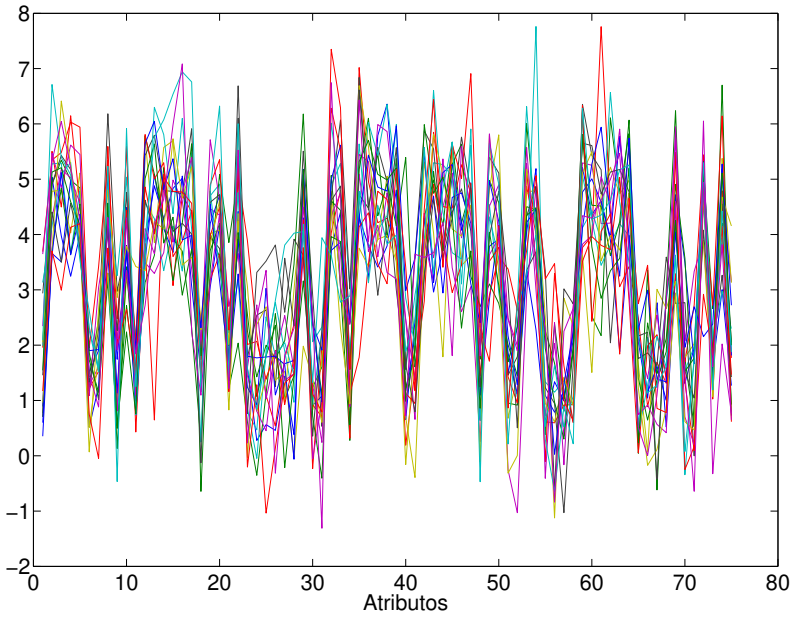


Fig. 4.25: Bicluster gerado para o caso de dados faltantes no atributo 80 da base *ArtDatasetMAR*. Cada linha do bicluster apresenta um comportamento similar em todos os atributos.

Nessa tabela, é possível perceber que os biclusters gerados contêm acima de 55% de seus atributos pertencentes àqueles atributos que definem os grupos com valores ausentes. Apesar de uma taxa razoável, vale verificar que o bicluster formado, levando em conta também seus objetos, consegue identificar um grupo bem definido levando facilmente ao motivo da ausência dos dados, conforme observado na Fig. 4.25 para o caso do atributo 80, que obteve a menor taxa. Essa informação ajuda a definir um grupo de objetos que pode ser utilizado de forma a aumentar a precisão da imputação de dados através das técnicas tradicionais.

Tab. 4.10: Resultados do experimento com o objetivo de descobrir o motivo dos dados faltantes do tipo MAR.

Atrib. Faltante	Atrib./Motivos	Atrib./Bic.	Atrib./Interseção	Taxa de Descobrimento
65	51	64	38	59, 38%
80	50	75	42	56, 00%
95	76	53	39	73, 58%

Base de Dados NMAR

A obtenção de informações para o tipo NMAR ocorre de uma forma diferente. Nesse caso, o objetivo inicial é descobrir objetos do mesmo grupo daqueles que contêm dados faltantes e, então, verificar quais os valores do atributo onde os dados estão ausentes. Para tanto, o procedimento inicia com um bicluster formado pelos objetos com ausência de valores e todas as colunas da base de dados. Em seguida, a coluna com maior valor de RQM é removida do bicluster até que sobre um número mínimo de colunas. Finalmente, as linhas da base de dados que não pertencem ao bicluster são adicionadas caso não ocorra aumento no RQM do bicluster.

Verifica-se, então, se os valores existentes no atributo com ausência de dados desse bicluster estão dentro da regra de geração definida no Apêndice A. Para isso, será contado o número de valores maior ou igual a $4,5$ no atributo 80 e os valores entre 1 e $2,5$ no atributo 65. Essas faixas de valores são aquelas determinadas na regra de geração da base de dados *ArtDatasetNMAR* como os motivos para a ausência de dados nos atributos correspondentes. Essa quantidade será comparada com o número de valores contidos no bicluster para esses atributos, verificando então a porcentagem de objetos que fazem parte do grupo e que são candidatos a terem seus valores ausentes.

Para exemplificar a análise dos resultados, suponha que o bicluster encontrado contenha 10 objetos, definidos pelo conjunto $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Desses objetos, apenas aqueles pertencentes ao subconjunto $\{1, 3, 4, 6, 8, 9\}$, com seis objetos, possuem valores dentro da faixa que determina o motivo da ausência de valor no atributo estudado. Consequentemente, pode-se dizer que esse bicluster obteve uma taxa de acerto de 60% na identificação dos objetos que poderiam ter seus valores ausentes naquele atributo.

Os resultados apresentados na Tab. 4.11 indicam quantos dos objetos pertencentes ao bicluster têm valores que os colocam dentro da faixa que ocasionou o motivo da ausência dos dados. A primeira coluna mostra o atributo que tem o valor faltante, a segunda coluna o número de objetos no bicluster, a terceira coluna mostra quantos desses objetos têm o atributo da primeira coluna dentro da faixa de valores estipulada na regra da ausência de dados, e a última coluna mostra o percentual desses objetos. A taxa de acerto do bicluster foi de $84,35\%$ para o primeiro motivo e $96,97\%$ para o segundo motivo, ou seja, a grande maioria dos objetos pertencentes ao bicluster está dentro da faixa de valores do motivo da ausência. Essa informação poderia ser utilizada para imputar os valores de forma mais precisa, levando em conta apenas essa faixa de valores definida pelo bicluster.

Esses dois experimentos mostram outra característica importante das técnicas de biclusterização: a capacidade de situar os objetos, mesmo com dados faltantes, em seus respectivos grupos, de forma a permitir não só uma análise de informação independente da imprecisão dos dados, mas também facilitar a utilização de outras técnicas para análise e/ou imputação dos dados. Em uma situação real possivelmente não existiria a informação se os dados faltantes seriam do tipo MAR, NMAR ou

Tab. 4.11: Resultados do experimento com o objetivo de descobrir o motivo dos dados faltantes do tipo NMAR.

Atributo Faltante	Objetos/Bicluster	Objetos/Candidatos	Taxa de Acerto
30	115	97	$84,35\%$
85	66	64	$96,97\%$

MCAR, porém, o conjunto de biclusters possivelmente conteria essa informação, sendo necessário ainda ser definida uma metodologia para identificá-los.

4.8 Síntese do capítulo

Este capítulo apresentou um novo método de δ-biclusterização, denominado *SwarmBcluster* por conta de sua inspiração em inteligência de enxame. Esse método consiste em duas partes: a primeira parte sendo uma heurística construtiva para encontrar um único bicluster, dado um ponto inicial na região da base de dados de forma a maximizar o seu volume; e a segunda parte sendo uma forma de gerar pontos iniciais para essa heurística, de modo a encontrar biclusters com baixa sobreposição e que maximizem a cobertura dos elementos da base de dados.

Uma vez que a heurística construtiva depende da ordem em que os elementos são inseridos, e não existe uma forma óbvia de determinar essa ordem, a meta-heurística conhecida como *Otimização por Colônia de Formigas* foi aplicada para cada fase de construção de soluções, de forma a obter o melhor bicluster possível dado um ponto inicial.

Esse método foi então comparado com outros métodos de δ-biclusterização em duas bases bem conhecidas da área de biclusterização e de microarranjos de DNA. Os resultados apontaram que o *SwarmBcluster* apresenta um melhor custo-benefício em relação aos outros métodos, tendo um melhor desempenho na maioria dos critérios de comparação e, principalmente, na cobertura dos dados. Testes estatísticos foram aplicados e analisados para confirmar o desempenho do *SwarmBcluster*. Em relação ao custo computacional, o *SwarmBcluster* se mostrou competitivo em sua configuração mais simples, sendo apenas um pouco pior do que a heurística construtiva de *Cheng & Church*. Isso aponta para uma capacidade de obter um conjunto de biclusters superior aos demais, sem ter um aumento do custo computacional.

Outra característica importante a ser notada é que o volume médio dos biclusters encontrados pelo *SwarmBcluster* pode ser aumentado com ajustes nos parâmetros. No entanto, esse aumento do volume médio vem acompanhado de um aumento significativo no custo computacional, o que na maioria dos casos pode não ser desejável.

Por fim, alguns testes complementares foram feitos para mostrar algumas características das técnicas de biclusterização, em específico o *SwarmBcluster*. Um experimento mostrou a capacidade dessas técnicas em eliminar parte do ruído existente na base de dados, uma vez que consideram um modelo de coerência aditiva perfeita. Outra característica é a capacidade de encontrar grupos de objetos e atributos que se comportam da mesma forma em torno de dados ausentes na base, permitindo assim uma extração de conhecimento sobre o motivo dessa ausência. Com esses resultados, percebe-se que a biclusterização é capaz de realizar as tarefas de seleção de atributos, processamento de sinais e aprendizado de máquina simultaneamente de acordo com o problema estudado e as medidas de correlação utilizadas para definir os biclusters. Os resultados desses experimentos fazem parte da definição da metodologia de imputação de dados a partir de biclusterização, apresentada no próximo capítulo.

Capítulo 5

Biclusterização e dados faltantes

O segundo capítulo dessa tese descreveu um problema enfrentado por diversos algoritmos de análise de dados: a incerteza dos valores. Essas incertezas podem aparecer na forma de ruído ou na simples ausência desses valores. Naquele mesmo capítulo, foi mostrada, através de uma base de dados artificial, a influência que essas incertezas têm na análise de dados, bem como formas simples para restituir os dados faltantes. Os métodos referenciados naquele capítulo, e muitos outros métodos, dependem principalmente da presença de objetos ou atributos que possuam um certo grau de similaridade.

Conforme também já apontado, mesmo com uma base de dados livre de qualquer incerteza, grande parte dos métodos utilizados para classificação e análise de dados não consegue perceber uma correlação parcial, ou seja, correlação entre um subconjunto de objetos que ocorre apenas em parte dos atributos. Essas correlações parciais podem ser encontradas através de técnicas de biclusterização, que procuram subconjuntos de objetos e atributos inter-relacionados.

No capítulo anterior, em uma das aplicações ilustrativas, apresentada na seção 4.7.4, foi mostrada a propriedade das técnicas de biclusterização de encontrar relações entre grupos de objetos mesmo diante de ruído. Porém, em se tratando de valores faltantes na base de dados, as técnicas de biclusterização, da forma em que foram propostas, podem gerar sub-matrizes com uma taxa muito alta de valores faltantes, tornando a análise impossível de ser realizada.

Dessa forma, é necessário criar uma metodologia para encontrar um conjunto de biclusters, mesmo diante de ausência de dados na base, reduzindo o impacto dessa ausência no cálculo da qualidade de cada bicluster. Dispondo desses biclusters com dados faltantes, é possível utilizar as propriedades de cada um desses biclusters para imputar tais dados faltantes, ou permitir uma análise estatística que conduza aos motivos que sustentam a falta de dados.

Para tanto, neste capítulo foi criada tal metodologia, procurando encontrar uma abordagem que cause o menor impacto possível na aplicação do algoritmo *SwarmBcluster* proposto no Cap. 4. Em

seguida, será demonstrado que a tarefa de imputação de um bicluster com ausência de dados pode ser transformada em um problema de otimização quadrática que, quando resolvido, contenha em seu vetor-solução aproximações melhores dos valores faltantes do que outras técnicas existentes.

5.1 Biclusterização de bases de dados com valores faltantes

Em técnicas de clusterização, dependendo da medida de similaridade utilizada, os valores faltantes podem ser ignorados quando do cálculo da proximidade entre duas amostras. Ao fazer isso, essas técnicas falham em averiguar possíveis influências dos atributos faltantes ao medir uma relação. Por outro lado, por definirem um modelo global e levarem em conta todos os atributos para definir a relação entre dois objetos, essas abordagens são capazes de medir a similaridade, mesmo que com perda de informação, na ausência de alguns desses valores, o que nem sempre é o caso na biclusterização, que efetua o cálculo em apenas um subconjunto definido dos atributos.

Conforme visto nos capítulos anteriores, a maioria das técnicas de biclusterização baseadas em coerência aditiva utiliza a fórmula de RQM para quantificar a qualidade de um bicluster. Essa quantificação é necessária ao processo heurístico para determinar as regiões promissoras do espaço de busca. Quando lidam com ausência de valores em uma base de dados, as técnicas de biclusterização enfrentam dois problemas: o aumento da proporção de valores ausentes em relação aos valores completos, devido à redução do espaço, provocando uma redução na confiabilidade do cálculo de similaridade; e a alteração nos valores estatísticos, das médias das linhas e colunas, causando uma alteração no valor real de RQM, o que é intensificado por ser uma medida de erro quadrático.

Por esse motivo, é necessária a criação de mecanismos capazes de estimar o RQM de um bicluster com valores faltantes e, ao mesmo tempo, restringir a quantidade desses valores dentro do bicluster. Numa solução proposta em de Castro et al. (2007a) e repetida em Coelho et al. (2009b), os valores faltantes eram simplesmente ignorados no cálculo do RQM e era acrescentada uma restrição à função-objetivo, tornando infactível todo bicluster com uma porcentagem de valores faltantes maior que um limiar. Apesar de encontrarem biclusters de boa qualidade, essa abordagem não explora efetivamente o conceito da biclusterização, pois, ao ignorar os possíveis valores ausentes, pode-se acabar inserindo ou removendo erroneamente uma linha ou coluna dentro de um bicluster.

Outro estudo de imputação utilizando biclusterização foi feita em (Colantonio et al., 2010) onde o problema foi restrito a matrizes binárias. Para estimar os valores o algoritmo proposto, denominado *ABBA*, buscava biclusters com uma grande quantidade de elementos com valor 1 e, todos os outros elementos desse bicluster eram estimados como também tendo valor 1.

Lembrando do experimento complementar feito no Cap. 4, as técnicas de biclusterização apresentam robustez quanto ao ruído dentro da base de dados, ou seja, utilizando a medida de RQM essas

técnicas são capazes de encontrar biclusters coerentes mesmo na presença de ruído, se o limiar residual for ajustado adequadamente. Levando em conta essa característica, uma forma de estimar o RQM na ausência de valores seria realizar uma pré-imputação na base de dados, utilizando alguma técnica simples de imputação, mesmo que imprecisa. O efeito prático é a introdução de ruído na base de dados. Utilizando essa base ruidosa, é possível ajustar o limiar de RQM e aplicar uma técnica de δ-biclusterização para encontrar biclusters coerentes que, apesar dos valores ruidosos, ofereçam uma melhor aproximação da base real do que simplesmente ignorar os valores ausentes.

Uma vez que tenham sido encontrados biclusters coerentes dentro dessa base ruidosa, os valores pré-imputados podem então ser novamente removidos da base e, utilizando a informação contida no modelo definido por cada bicluster, encontrar novos valores, mais próximos aos valores reais, dada a capacidade que o bicluster possui em reduzir o ruído, que nesse caso é o erro de predição.

5.2 Estimação de dados faltantes em um bicluster

Após obter o conjunto de biclusters da base de dados pré-imputada, é necessário definir um procedimento para reimputar tais valores, reduzindo o ruído inserido, com base nas informações contidas no modelo gerado pelo bicluster. O modelo definido pela coerência aditiva considera que os elementos dos biclusters podem ser descritos através das médias das linhas e das colunas, e a qualidade desses biclusters é mensurada através do erro quadrático médio entre modelo de coerência e seus valores reais. Portanto, pode-se afirmar que uma boa aproximação para os valores ausentes é aquela que minimiza esse erro quadrático. Intuitivamente, é possível pensar que o simples cálculo desses valores poderia ser feito utilizando as médias do bicluster, com valores pré-imputados, através da Eq. 3.19, reproduzida aqui:

$$b_{ij} = b_{Ij} + b_{iJ} - b_{IJ}, \tag{5.1}$$

onde b_{ij} é o elemento do bicluster a ser imputado, b_{Ij} é a média da coluna j do bicluster com valores pré-imputados, b_{iJ} a média da linha i e b_{IJ} a média total do bicluster. Porém, é necessário lembrar que, ao calcular o modelo utilizando valores ruidosos, esse próprio modelo contém ruído que será inserido nos valores imputados através da fórmula acima. Para reduzir parte desse ruído, é necessário encontrar valores que, ao serem imputados dentro desse bicluster, minimizam o erro quadrático médio entre o bicluster contendo esses novos valores e o modelo definido por ele (de França et al., 2009).

Minimizar a equação de RQM, com os valores ausentes como variáveis, é equivalente a resolver um problema de otimização quadrática que, em condições ideais, pode ser resolvido em tempo polinomial. Para tanto, primeiramente a equação de RQM deve ser apresentada na forma convencional de um problema de otimização quadrática, para que seja possível o uso de algoritmos tradicionais na

$$\mathrm{B} = \begin{bmatrix} 1 & 2 & - & 4 \\ 5 & - & 7 & 8 \\ 9 & 10 & 11 & - \end{bmatrix} \quad \mathrm{B} = \begin{bmatrix} 1 & 2 & x_1 & 4 \\ 5 & x_2 & 7 & 8 \\ 9 & 10 & 11 & x_3 \end{bmatrix}$$

(a) (b)

Fig. 5.1: Um exemplo de bicluster coerente com valores faltantes (a), que serão convertidos em variáveis a serem otimizadas (b).

obtenção da solução do problema. Em seguida, é necessário verificar se a obtenção da solução por métodos de complexidade polinomiais é possível, ou seja, se as características do bicluster permitem a aplicação dessas técnicas.

As próximas subseções tratam da formulação do problema de otimização, particularmente abordando o problema de transformar um bicluster com valores faltantes em um problema quadrático, e indicando quais as características que o bicluster deve possuir para garantir uma solução em tempo polinomial.

5.2.1 Formulação quadrática

Tomando como exemplo o bicluster com valores faltantes apresentado na Fig. 5.1(a), os valores ausentes dessa matriz são então transformados em variáveis de um problema de otimização, conforme a Fig. 5.1(b). Com isso, o bicluster passa a conter 3 variáveis a serem definidas. Calculando o RQM desse bicluster levando em conta as variáveis, obtém-se:

$$\begin{aligned} b_{iJ} &= \left\{ \frac{7 + x_1}{4}, \frac{20 + x_2}{4}, \frac{30 + x_3}{4} \right\} \\ b_{Ij} &= \left\{ 5, \frac{12 + x_2}{3}, \frac{18 + x_1}{3}, \frac{12 + x_3}{3} \right\} \\ b_{IJ} &= \frac{57 + x_1 + x_2 + x_3}{12} \\ RQM &= \frac{x_1^2}{24} + \frac{x_1 \cdot x_2}{72} + \frac{x_1 \cdot x_3}{72} - \frac{x_1}{2} + \frac{x_2^2}{24} + \frac{x_2 \cdot x_3}{72} - \frac{17x_2}{24} + \frac{x_3^2}{24} - \frac{9x_3}{8} + \frac{77}{8}, \end{aligned} \tag{5.2}$$

que configura um problema de otimização quadrática com 3 variáveis. Os problemas quadráticos são habitualmente descritos da seguinte forma (Goldfarb & Idnani, 1983):

$$\begin{aligned} \text{Min } H_{I,J}(x) &= \frac{1}{2}x^T Qx + b^T x + c, \\ \text{s.a. } l_i &\leq x_i \leq u_i, i = 1, ..., n. \end{aligned} \tag{5.3}$$

onde $x = [x_1, x_2, ..., x_n]^T$, $Q \in \Re^{n \times n}$, $b \in \Re^n$, $c \in \Re$ e l_i e u_i são os limitantes inferiores e superiores da variável i, respectivamente. Com a fórmula de RQM descrita na Eq. 5.2, é possível obter facilmente a matriz Q, o vetor b e o escalar c e, a partir deles, resolver o problema utilizando algum algoritmo especializado (Goldfarb & Idnani, 1983).

Porém, para que seja possível resolver esse problema computacionalmente, e de forma genérica para qualquer bicluster com valores faltantes, é necessário encontrar fórmulas fechadas para obter os valores de Q, b e c. Estas fórmulas fechadas foram desenvolvidas nesta tese e serão descritas a seguir. Relembrando a fórmula do RQM:

$$H_{I,J}(x) = \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} r_{ij}^2(x) = \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (a_{ij}(x) - a_{Ij}(x) - a_{iJ}(x) + a_{IJ}(x))^2, \tag{5.4}$$

onde $H_{I,J}(x)$ é o RQM do bicluster $B = A(I, J)$, levando em conta o vetor de variáveis x, e $|\zeta|$ expressa a quantidade de elementos no conjunto ζ. Sabendo que a matriz Q é obtida calculando a matriz hessiana, o vetor b é o vetor gradiente quando $x = 0$ e c é o escalar resultante quando se faz $H_{I,J}(0)$, é possível obter uma equação fechada e genérica para cada um desses elementos, com base nos vetores gradientes e na matriz hessiana, através das derivadas primeira e segunda da Eq. 5.4.

O vetor gradiente de $H_{I,J}(x)$, dado pela Eq. 5.3 pode ser calculado por:

$$\nabla H_{I,J}(x) = Qx + b, \tag{5.5}$$

com $\nabla H_{I,J} \in \Re^n$ e $\nabla_k H_{I,J}$ denotando o k-ésimo elemento de $\nabla H_{I,J}$ ($k = 1, ..., n$).

Considerando agora o cálculo do RQM dado na Eq. 5.4, obtém-se:

$$\nabla_k H_{I,J}(x) = \frac{\partial}{\partial x_k} \left[\frac{1}{n'm'} \cdot \sum_{i,j \in I,J} r_{ij}^2(x) \right], \tag{5.6}$$

$$\nabla_k H_{I,J}(x) = \frac{1}{n'm'} \cdot \sum_{i,j \in I,J} \frac{\partial r_{ij}^2(x)}{\partial x_k}, \tag{5.7}$$

$$\nabla_k H_{I,J}(x) = \frac{2}{n'm'} \cdot \left[\sum_{i,j \in I,J} r_{ij}(x) \cdot \frac{\partial r_{ij}(x)}{\partial x_k} \right], \tag{5.8}$$

onde r_{ij} é o valor do resíduo do elemento da linha i e coluna j no bicluster, e n' e m' são as quantidades

de linhas ($|I|$) e colunas ($|J|$), respectivamente, do bicluster.

Cada elemento (k, l) da matriz hessiana do resíduo quadrático médio pode ser calculado por:

$$\nabla^2_{k,l} H_{I,J}(x) = \frac{2}{n'm'} \cdot \frac{\partial}{\partial x_l} \left[\sum_{i,j \in I,J} r_{ij}(x) \cdot \frac{\partial r_{ij}(x)}{\partial x_k} \right], \tag{5.9}$$

$$\begin{aligned} \nabla^2_{k,l} H_{I,J}(x) = \frac{2}{n'm'} \times \\ \times \sum_{i,j \in I,J} \left[\frac{\partial r_{ij}(x)}{\partial x_l} \cdot \frac{\partial r_{ij}(x)}{\partial x_k} + r_{ij}(x) \cdot \frac{\partial^2 r_{ij}(x)}{\partial x_l \partial x_k} \right]. \end{aligned} \tag{5.10}$$

Como $r_{ij}(x)$ é uma equação de primeiro grau, então $\frac{\partial^2 r_{ij}(x)}{\partial x_l \partial x_k} = 0$ e a equação assume a forma:

$$\nabla^2_{k,l} H_{I,J}(x) = \frac{2}{n'm'} \cdot \sum_{i,j \in I,J} \left[\frac{\partial r_{ij}(x)}{\partial x_l} \cdot \frac{\partial r_{ij}(x)}{\partial x_k} \right]. \tag{5.11}$$

Consequentemente, uma vez que, $H_{I,J}(x) = \frac{1}{2} x^T Q x + b^T x + c$, os valores dos elementos $q_{k,l}$ de Q ($k = 1, ..., n$ e $l = 1, ..., n$) serão:

$$q_{k,l} = \nabla^2_{k,l} H_{I,J}(x) = \frac{2}{n'm'} \cdot \sum_{i,j \in I,J} \left[\frac{\partial r_{ij}(x)}{\partial x_l} \cdot \frac{\partial r_{ij}(x)}{\partial x_k} \right], \tag{5.12}$$

e dos elementos b_k de b serão:

$$b_k = \nabla_k H_{I,J}(0) = \frac{2}{n'm'} \cdot \left[\sum_{i,j \in I,J} r_{ij}(0) \cdot \frac{\partial r_{ij}(0)}{\partial x_k} \right]. \tag{5.13}$$

Portanto, para calcular tanto a matriz Q quanto o vetor b, é necessário obter os valores das derivadas parciais de $r_{ij}(x)$ em relação a x_k ($k = 1, ..., n$). Essas derivadas parciais podem assumir um entre quatro valores distintos, de acordo com a posição relativa a cada variável.

Retomando a equação de resíduo (Eq. 5.4) e a Fig. 5.1, no cálculo de $r_{i,j}$, por exemplo, uma variável x_i que se encontra nessa mesma posição do bicluster estará presente em todos os termos da equação, pois fará parte de todas as médias e do valor de $a_{ij}(x)$. Já no caso em que essa variável não se encontrar nem na linha i e nem na coluna j, ela estará presente apenas no termo referente à média total do bicluster. Analogamente, quando a variável estiver localizada apenas na linha i ou apenas na coluna j, ela estará presente tanto na média total como na média da linha ou da coluna, conforme o caso. Como cada termo dessa equação é o próprio elemento ou o somatório de elementos da linha, coluna ou do bicluster inteiro, divididos pelo número de elementos nesse somatório, a derivada em cada um desses termos será essa própria constante, sempre que a variável aparecer no somatório.

Portanto, o valor da derivada em cada uma das quatro situações descritas é:

$$\frac{\partial r_{ij}}{\partial x_k} = \begin{cases} c_1 = 1 - \frac{1}{n'} - \frac{1}{m'} + \frac{1}{n'm'} = \frac{(n'-1)\cdot(m'-1)}{n'm'}, & \text{se } i = k_i, j = k_j \\ c_2 = -\frac{1}{m'} + \frac{1}{n'm'} = \frac{(1-n')}{n'm'}, & \text{se } i = k_i, j \neq k_j \\ c_3 = -\frac{1}{n'} + \frac{1}{n'm'} = \frac{(1-m')}{n'm'}, & \text{se } j = k_j, i \neq k_i \\ c_4 = \frac{1}{n'm'}, & \text{se } i \neq k_i, j \neq k_j \end{cases}, \tag{5.14}$$

onde k_i e k_j dizem respeito à posição da linha (i) e da coluna (j) da variável k no bicluster.

Os cálculos de Q e b podem ser simplificados pela substituição dos valores da derivada. No caso do vetor b, devemos enumerar dentro do somatório a ocorrência de cada derivada da Eq. 5.14, sabendo que c_1 ocorre apenas uma vez, quando derivando na posição da variável, c_2 ocorre em toda a linha onde a variável se encontra, exceto na coluna da própria variável, c_3 em toda a coluna da variável, exceto no cruzamento dessa coluna com sua linha, e c_4 ocorre em todos os outros casos. Essas enumerações levam a:

$$\begin{aligned} \nabla_k H_{I,J}(0) = \frac{2}{n'm'}\Bigg[& c_1 \cdot r(0)_{k_i,k_j} + c_2 \cdot \sum_{j \in m'} r(0)_{k_i,j} - c_2 \cdot r(0)_{k_i,k_j} + c_3 \cdot \sum_{i \in n'} r(0)_{i,k_j} - \\ & - c_3 \cdot r(0)_{k_i,k_j} + c_4 \cdot \sum_{i,j \in I,J} r(0)_{i,j} - c_4 \cdot \sum_{j \in m'} r(0)_{k_i,j} - c_4 \cdot \sum_{i \in n'} r(0)_{i,k_j} + \\ & + c_4 \cdot r(0)_{k_i,k_j} \Bigg] \end{aligned} \tag{5.15}$$

$$\begin{aligned} \nabla_k H_{I,J}(0) = \frac{2}{n'm'}\Bigg[& (c_1 - c_2 - c_3 + c_4) \cdot r(0)_{k_i,k_j} + (c_2 - c_4) \cdot \sum_{j \in m'} r(0)_{k_i,j} + \\ & + (c_3 - c_4) \cdot \sum_{i \in n'} r(0)_{i,k_j} + c_4 \cdot \sum_{i,j \in I,J} r(0)_{i,j} \Bigg], \end{aligned} \tag{5.16}$$

como

$$c_1 - c_2 - c_3 + c_4 = 1, \tag{5.17}$$

e

$$c_2 - c_4 = -\frac{1}{m'} \tag{5.18}$$

$$c_3 - c_4 = -\frac{1}{n'}, \tag{5.19}$$

então

$$\nabla_k H_{I,J}(0) = \frac{2}{n'm'}\left[r(0)_{k_i,k_j} - \frac{1}{m'}\cdot\sum_{j\in m'} r(0)_{k_i,j} - \frac{1}{n'}\cdot\sum_{i\in n'} r(0)_{i,k_j} + \frac{1}{n'm'}\cdot\sum_{i,j\in I,J} r(0)_{i,j}\right] \tag{5.20}$$

$$\nabla_k H_{I,J}(0) = \frac{2}{n'm'}\left[r(0)_{k_i,k_j} - r(0)_{k_i,J} - r(0)_{I,k_j} + r(0)_{I,J}\right]. \tag{5.21}$$

Expandindo cada membro da equação na forma:

$$\begin{aligned} r(0)_{k_i,J} &= \frac{1}{m'}\cdot\sum_{j\in J}(b_{k_i,j} - b_{k_i,J} - b_{I,j} + b_{I,J}) \\ &= b_{k_i,J} - b_{k_i,J} - b_{I,J} + b_{I,J} \\ &= 0, \end{aligned} \tag{5.22}$$

$$\begin{aligned} r(0)_{I,k_j} &= \frac{1}{n'}\cdot\sum_{i\in I}(b_{i,k_j} - b_{i,J} - b_{I,k_j} + b_{I,J}) \\ &= b_{I,k_j} - b_{I,J} - b_{I,k_j} + b_{I,J} \\ &= 0, \end{aligned} \tag{5.23}$$

$$\begin{aligned} r(0)_{I,J} &= \frac{1}{n'm'}\cdot\sum_{i,j\in I,J}(b_{i,j} - b_{i,J} - b_{I,j} + b_{I,J}) \\ &= b_{I,J} - b_{I,J} - b_{I,J} + b_{I,J} \\ &= 0, \end{aligned} \tag{5.24}$$

$$\begin{aligned} r(0)_{k_i,k_j} &= b_{k_i,k_j} - b_{k_i,J} - b_{I,k_j} + b_{I,J} \\ &= b_{I,J} - b_{k_i,J} - b_{I,k_j}, \end{aligned} \tag{5.25}$$

resulta:

$$\nabla_k H_{I,J}(0) = \frac{2}{n'm'}\left[b_{I,J} - b_{k_i,J} - b_{I,k_j}\right]. \tag{5.26}$$

Dessa forma, torna-se desnecessário calcular o resíduo quadrático médio para obter o vetor b, bastando apenas calcular as médias dos valores das linhas e das colunas e do bicluster inteiro, substituindo os valores das variáveis por zero.

Da mesma forma, para calcular os valores da matriz Q, primeiro é necessário enumerar quanto de cada um dos quatro valores de derivadas possíveis aparece em cada elemento da matriz. Considere o bicluster da Fig. 5.2(a), contendo duas variáveis. Calcula-se a derivada da variável x_1 em relação a cada elemento do bicluster, resultando na matriz da Fig. 5.2(b). Similarmente, na Fig. 5.2(c), é ilustrado o cálculo das derivadas para a variável x_2. Para determinar q_{x_k,x_l}, conforme Eq. 5.11, basta multiplicar os elementos das duas matrizes entre si e somar os valores da matriz resultante.

$$\underset{\text{(a)}}{\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & x_1 & 7 & x_2 \\ 9 & 10 & 11 & 12 \end{bmatrix}} \quad \underset{\text{(b)}}{\begin{bmatrix} c_4 & c_3 & c_4 & c_4 \\ c_2 & c_1 & c_2 & c_2 \\ c_4 & c_3 & c_4 & c_4 \end{bmatrix}} \quad \underset{\text{(c)}}{\begin{bmatrix} c_4 & c_4 & c_4 & c_3 \\ c_2 & c_2 & c_2 & c_1 \\ c_4 & c_4 & c_4 & c_3 \end{bmatrix}}$$

Fig. 5.2: Exemplo ilustrativo para o cálculo da matriz hessiana: (a) bicluster com duas variáveis; (b) derivadas em relação à variável x_1; (c) derivadas em relação à variável x_2.

Os valores a serem multiplicados entre si podem ser descobertos através da posição relativa às duas variáveis; no caso das variáveis x_k e x_l, às posições (k_i, k_j) e (l_i, l_j) no bicluster, respectivamente. O valor de cada elemento da hessiana pode ser determinado, então, em função da posição relativa entre as duas variáveis em questão, conforme detalhado a seguir.

(a) valor das diagonais q_{x_k,x_k}:

Nesse caso, a hessiana é calculada pela multiplicação dos valores da mesma variável e sabendo que a matriz de derivadas da variável contém uma ocorrência de c_1, $(n'-1)$ ocorrências de c_2, $(m'-1)$ ocorrências de c_3 e o restante dos elementos iguais a c_4. Com isso, o valor da hessiana nos elementos da diagonal fica:

$$q_{x_k,x_k} = c_1^2 + (n'-1)\cdot c_3^2 + (m'-1)\cdot c_2^2 + (n'm' - n' - m' + 1)\cdot c_4^2, \tag{5.27}$$

que, substituindo pelos valores da Eq. 5.14 produz:

$$q_{x_k,x_k} = \frac{n'^2m'^2 - n'm'^2 - n'^2m' + n'm'}{n'^2m'^2}. \tag{5.28}$$

Simplificando, obtém-se:

$$q_{x_k,x_k} = \frac{(n'-1)\cdot(m'-1)}{n'm'} = c_1. \tag{5.29}$$

(b) valor de $q_{x_k,x_l,k_j\neq l_j}$:

Para o caso em que $k_i = l_i$ e $k_j \neq l_j$, ou seja, quando as duas variáveis estão localizadas na mesma linha, mas em colunas diferentes, temos:

$$q_{x_k,x_l} = 2\cdot c_1\cdot c_2 + (m'-2)\cdot c_2^2 + 2\cdot(n'-1)\cdot c_3\cdot c_4 + (n'm' + 2 - m' - 2n')\cdot c_4^2, \tag{5.30}$$

que, substituindo os valores da Eq. 5.14 e simplificando, produz:

$$q_{x_k,x_l} = \frac{(1-n')}{n'm'} = c_2. \tag{5.31}$$

(c) valor de $q_{x_k,x_l,k_i\neq l_i}$:

Analogamente, para o caso em que $k_i \neq l_i$ e $k_j = l_j$, ou seja, quando as duas variáveis estão localizadas na mesma coluna, mas em linhas diferentes, temos:

$$q_{x_k,x_l} = 2\cdot c_1\cdot c_3 + 2\cdot(m'-1)\cdot c_2\cdot c_4 + (n'-2)\cdot c_3^2 + (n'm' + 2 - 2m' - n')\cdot c_4^2, \tag{5.32}$$

que, substituindo os valores da Eq. 5.14 e simplificando, produz:

$$q_{x_k,x_l} = \frac{(1-m')}{n'm'} = c_3. \tag{5.33}$$

(d) valor de $q_{x_k,x_l,k_i\neq l_i,k_j\neq l_j}$:

Finalmente, para $k_i \neq l_i$ e $k_j \neq l_j$, ou seja, quando as duas variáveis estão localizadas em linhas e colunas distintas do bicluster, temos:

$$q_{x_k,x_l} = 2\cdot c_1\cdot c_4 + 2\cdot(m'-2)\cdot c_2\cdot c_4 + 2\cdot(n'-2)\cdot c_3\cdot c_4 + 2\cdot c_2\cdot c_3 + (n'm' + 4 - 2n' - 2m')\cdot c_4^2, \tag{5.34}$$

que, substituindo os valores da Eq. 5.14 e simplificando, produz:

$$q_{x_k,x_l} = \frac{1}{n'm'} = c_4. \tag{5.35}$$

A partir dos resultados obtidos acima, é possível concluir que:

$$\sum_{i,j \in I,J} \left[\frac{\partial r_{ij}(x)}{\partial x_k} \cdot \frac{\partial r_{ij}(x)}{\partial x_l} \right] = \frac{\partial r_{k_i k_j}}{\partial x_l} = \frac{\partial r_{l_i l_j}}{\partial x_k}, \tag{5.36}$$

que é de fácil obtenção através da Eq. 5.14, permitindo o cálculo da matriz Q e do vetor b. O valor de c pode ser obtido calculando $H_{I,J}(0)$, ou seja, calculando o RQM do bicluster atribuindo valor 0 às variáveis.

5.2.2 Garantia de solução

Com Q e b calculados, para tornar possível a solução do problema quadrático em tempo polinomial, deve-se garantir que a matriz Q seja definida positiva, para que exista a inversa dessa matriz e exista um ótimo global finito. Oportunamente, isso pode ser garantido através do controle entre o volume do bicluster e a taxa de dados faltantes.

Teorema 5.2.1 *Dado um bicluster B, contendo n′ linhas e m′ colunas, e r sendo a maior taxa de valores faltantes nas linhas e nas colunas desse bicluster. Então é condição suficiente que* $r < \frac{n'm'-2}{3n'm'-2n'-2m'}$ *para que a matriz hessiana Q, calculada a partir desse bicluster, seja definida positiva.*

Prova: Uma forma de verificar se uma dada matriz quadrada e simétrica é definida positiva é através da seguinte inequação:

$$q_{i,i} > \sum_{j \neq i} |q_{i,j}|, \tag{5.37}$$

ou seja, se todos os elementos da diagonal da matriz são positivos e o somatório do valor absoluto dos elementos de qualquer linha, excetuando-se o elemento da diagonal, for menor que o elemento da diagonal, então a matriz é definida positiva (Harville, 2001, 2008).

No caso estudado, o valor dos elementos da diagonal é sempre igual a c_1 e, portanto, positivo e não nulo para $n, m > 1$.

Supondo uma restrição de número de valores ausentes por linha e coluna, limitando a quantidade de ausência de dados em $r \cdot m'$ elementos em cada linha e $r \cdot n'$ elementos em cada coluna, com $r \in \Re$ e $0 < r < 1$, e levando em conta que cada elemento ausente é uma variável do problema, temos que o somatório dos valores absolutos dos elementos de cada linha, excetuando o elemento da diagonal, terá um valor máximo de:

$$max(\sum_{j=1,j\neq i}^{vars} |q_{i,j}|) = (r \cdot m' - 1) \cdot |c_2| + (r \cdot n' - 1) \cdot |c_3| + (r \cdot n'm' - r \cdot n' - r \cdot m' + 1) \cdot |c_4|, \quad (5.38)$$

onde *vars* é o número de variáveis do problema. Substituindo os valores da Eq. 5.14, obtém-se:

$$max(\sum_{j=1,j\neq i}^{vars} |q_{i,j}|) = (r \cdot m' - 1) \cdot |\frac{1 - n'}{n'm'}| + (r \cdot n' - 1) \cdot |\frac{1 - m'}{n'm'}| + \quad (5.39)$$
$$+ (r \cdot n'm' - r \cdot n' - r \cdot m' + 1) \cdot |\frac{1}{n'm'}|,$$

e, como $n', m' > 1$ faz com que $c_2, c_3 < 0$, calculando os valores absolutos da equação anterior resulta:

$$max(\sum_{j=1,j\neq i}^{vars} |q_{i,j}|) = (r \cdot m' - 1) \cdot \frac{n' - 1}{n'm'} + (r \cdot n' - 1) \cdot \frac{m' - 1}{n'm'} + \quad (5.40)$$
$$+ (r \cdot n'm' - r \cdot n' - r \cdot m' + 1) \cdot \frac{1}{n'm'}.$$

Simplificando a equação, obtém-se:

$$max(\sum_{j=1,j\neq i}^{vars} |q_{i,j}|) = \frac{3rn'm' - (2r + 1)n' - (2r + 1)m' + 3}{n'm'}. \quad (5.41)$$

Com isso, para que a matriz hessiana seja definida positiva, é necessário que

$$\frac{(n' - 1)(m' - 1)}{n'm'} > \frac{3rn'm' - (2r + 1)n' - (2r + 1)m' + 3}{n'm'}, \quad (5.42)$$

que resolvendo conduz a:

$$n'm' - n' - m' + 1 > 3rn'm' - (2r + 1)n' - (2r + 1)m' + 3,$$

$$(2r + 1)n' - n' + (2r + 1)m' - m' > 3rn'm' - n'm' + 2,$$

$$2rn' + 2rm' > 3rn'm' - n'm' + 2,$$

$$2r(n' + m') > 3rn'm' - n'm' + 2,$$

$$2r(n' + m') - 3rn'm' > -n'm' + 2,$$

$$3rn'm' - 2r(n' + m') < n'm' - 2,$$

$$r(3n'm' - 2n' - 2m') < n'm' - 2,$$

$$r < \frac{n'm' - 2}{3n'm' - 2n' - 2m'}. \tag{5.43}$$

Teorema 5.2.2 *Seja r a maior taxa de dados faltantes nas linhas e colunas do bicluster B. O maior valor que essa taxa pode assumir para que a matriz hessiana Q, correspondente, seja definida positiva é igual a* $0,5$ *quando* $n' = m' = 2$.

Prova: Para determinar a taxa máxima r, é necessário primeiro encontrar o sinal do termo do lado direito da inequação. Como $n', m' > 1$, e pertencentes aos números inteiros, então $n'm' - 2 > 0$ e verificando o denominador

$$3n'm' - 2n' - 2m' > 0,$$

$$3n'm' - 2m' > 2n',$$

$$(3n' - 2)m' > 2n',$$

$$m' > \frac{2n'}{(3n' - 2)}, \tag{5.44}$$

e, novamente, sabendo que $n', m' > 1$, é possível garantir que a inequação acima é válida quando

$$\frac{2n'}{(3n' - 2)} \leq 1,$$

$$2n' \leq 3n' - 2,$$

$$2 \leq n',$$

ou

$$n' \geq 2, \tag{5.45}$$

resultando em

$$m' \geq 1, \tag{5.46}$$

o que é factível segundo os domínios de n' e m'. Isso confirma os sinais da Eq. 5.43.

Uma vez que $n', m' > 1$ e $n', m' \in \mathbb{N}$, o menor valor de n' e m' é 2 e, portanto:

$$r < 0,5, \tag{5.47}$$

onde $0,5$ representa o maior valor de r que torna a matriz hessiana definida positiva em um bicluster mínimo.

Teorema 5.2.3 *Seja r a maior taxa de dados faltantes nas linhas e colunas do bicluster B. O menor limiar possível, que restringe essa taxa, para que a matriz hessiana Q, correspondente, seja definida positiva é igual a* $\frac{1}{3}$.

Prova: Aumentando o valor de n' e m', é possível obter um limitante inferior para o valor máximo de r:

$$\begin{aligned}
\lim_{n',m' \to +\infty} \frac{n'm' - 2}{3n'm' - 2n' - 2m'} &= \lim_{n',m' \to +\infty} \frac{1}{3} \cdot \frac{n'm' - 2}{n'm' - \frac{2}{3}(n' + m')} \\
&= \lim_{n',m' \to +\infty} \frac{1}{3} \cdot \left(\frac{n'm'}{n'm' - \frac{2}{3}(n' + m')} - \frac{2}{n'm' - \frac{2}{3}(n' + m')} \right) \\
&= \frac{1}{3}.
\end{aligned} \tag{5.48}$$

Portanto, para que a matriz hessiana do problema quadrático proposto seja definida positiva, e o problema tenha um único ótimo de mínimo global, é uma condição suficiente para qualquer bicluster que a taxa máxima de valores ausentes em cada linha e coluna do bicluster seja menor ou igual a $\frac{1}{3}$, e o limitante superior específico para cada bicluster pode ser calculado pela Eq. 5.43. Com isso torna-se possível a aplicação de algoritmos de otimização quadrática que conseguem obter a solução de ótimo global em tempo polinomial (Goldfarb & Idnani, 1983).

5.3 Algoritmo de biclusterização para imputação de dados

Para utilizar o algoritmo *SwarmBcluster* visando à imputação de dados, fazem-se necessárias algumas modificações em sua estrutura interna, de forma a garantir condições importantes como:

- efetuar a pré-imputação dos dados para que o algoritmo trabalhe com uma base completa;
- garantir que todos os valores ausentes sejam cobertos pelo conjunto de biclusters;
- garantir que cada bicluster respeite a restrição máxima de valores ausentes por linhas e colunas;
- assegurar que os biclusters gerados tenham uma qualidade mínima para obtenção de bons resultados.

O algoritmo modificado é ilustrado em Alg. 9 e explicado em seguida.

Algoritmo 9 Modificações do algoritmo *SwarmBcluster* para imputação de valores.

Impute valores ausentes utilizando uma metodologia simplificada;
Crie duas listas de candidatos, uma das linhas e colunas já cobertas e outra baseada nos valores faltantes;
enquanto existir valor ausente na lista de candidatos **faça**
    Escolha como bicluster inicial a linha que contém o próximo valor faltante e suas colunas não cobertas;
    Aplique o algoritmo ACO com essa linha inicial, impedindo a remoção da coluna que contém o valor faltante;
    se Bicluster gerado atende as restrições de máxima taxa de valores ausentes **então**
        Aplique um método de otimização quadrática e guarde os valores gerados sem imputá-los;
    senão
        Armazene o bicluster em memória;
    fim se
    Atualize as listas de candidatos de acordo com as variáveis ausentes calculadas e as linhas e colunas cobertas;
fim enquanto
Impute valores já calculados na base de dados;
Processe novamente biclusters armazenados em memória;

O algoritmo inicia o processo imputando os valores através de uma metodologia simples. Para o caso deste trabalho, foi utilizado o algoritmo *Vizinho mais Próximo*, explicado anteriormente. Quando não é possível encontrar um valor para determinado atributo, utiliza-se a mediana dos valores completos para a imputação. Em seguida, duas listas de candidatos são criadas: uma idêntica à proposta original do *SwarmBcluster* e outra contendo os valores ausentes que ainda não foram imputados por nenhum bicluster.

Dando sequência ao procedimento, enquanto ainda existir um valor ausente não coberto por nenhum bicluster, um desses elementos é escolhido e a linha que contém tal elemento será escolhida como ponto inicial do bicluster. O algoritmo ACO é aplicado para formação do bicluster, contando com o impedimento da remoção da coluna onde se encontra tal valor ausente, garantindo que ele será imputado.

Ao encontrar o bicluster, este é avaliado quanto à restrição de quantidade de valores faltantes em suas linhas e colunas, utilizando a Eq. 5.43. Caso a matriz hessiana gerada a partir do bicluster seja definida positiva, um algoritmo de otimização quadrática é aplicado para encontrar os valores faltantes nesse bicluster. Neste trabalho, foi utilizado o método proposto em Goldfarb & Idnani (1983) que resolve um problema quadrático convexo em $O(vars^3.L)$ operações aritméticas, onde $vars$ é o número de variáveis do problema, ou número de valores faltantes, e L é o comprimento da representação binária computacional de cada variável.

Caso o bicluster encontrado não obedeça aos critérios estabelecidos, ele é armazenado em memória para uso posterior. Note que os valores encontrados até então não são imputados imediatamente na base de dados, mas armazenados em separado até o final do processo, pois é possível que um dado valor ausente apareça em mais de um bicluster. Essa informação proveniente de múltiplos modelos, cada um vinculado a um bicluster, será utilizada para obter um valor mais preciso de imputação, ao final do processo.

Quando o processo é concluído, os valores já encontrados são então imputados na base. Caso um mesmo valor seja calculado por diferentes biclusters, o valor imputado será a média ponderada dos valores calculados. O peso dessa média, neste trabalho, foi definido como a razão entre o volume do bicluster e o número de valores ausentes dentro dele, sendo normalizados ao final do processo pelo somatório dos pesos. Após a imputação desses valores, aqueles biclusters cujas matrizes hessianas não eram definidas positivas são reprocessados, só que dessa vez levando em conta os valores já imputados, reduzindo a quantidade de valores ausentes da base de dados e, possivelmente, tornando a matriz hessiana correspondente definida positiva. Caso, após essas passagens, ainda exista algum valor não imputado, é utilizado o próprio valor gerado pela pré-imputação para preenchê-lo, pois supõe-se que não exista bicluster capaz de estimá-lo.

5.4 Resultados experimentais

Para verificar a qualidade na imputação dos dados através de técnicas de biclusterização, foram realizados experimentos em três bases de dados distintas. A primeira é a base *ArtDatasetRuido*, com dimensão 1.000×100, introduzida no Cap. 2, em que foi gerada inicialmente uma base de dados perfeita e, posteriormente, acrescentados ruídos a ela. A segunda base de dados é a *Yeast*, com

dimensão 2.884×17, já utilizada nos experimentos de biclusterização e que será utilizada aqui com casos do tipo MCAR, com diferentes taxas de dados faltantes.

Finalmente, a terceira base de dados utilizada foi a conhecida como *Jester*, que é uma base utilizada para testar algoritmos de Filtragem Colaborativa (Cap. 2). Um sistema composto por 101 piadas que apresenta algumas delas sequencialmente para cada um de seus 24.983 usuários, que atribuíram uma nota entre -10 e $+10$ para a piada da vez. A ordem em que as piadas eram apresentadas era inicialmente aleatória e, conforme o usuário atribuía notas, o sistema adaptava a ordem de apresentação de acordo com o gosto do usuário.

Uma vez que a aquisição dessa base foi feita utilizando usuários reais, ela não contém todos os seus valores. Por um lado, isso torna a base de dados mais próxima de uma situação real. Por outro, torna difícil comparar efetivamente o desempenho do algoritmo, uma vez que os valores não são conhecidos. Geralmente, para ser possível testar os algoritmos nesse caso, é feita a remoção aleatória de alguns valores conhecidos e é requisitado ao algoritmo que impute apenas esses valores. Mas isso causa uma limitação na quantidade de dados que podem ser retirados da base, uma vez que ela já conta com ausência de dados. Isso interfere também na estimativa adequada de desempenho dos algoritmos, uma vez que a ausência de dados em certa região pode prejudicar alguns dos métodos testados ou trazer maior vantagem para outros. Por isso, a base de dados *Jester* foi reduzida para uma base de dados completa contendo 7.199 usuários e 58 piadas, removendo todos os elementos faltantes.

Para verificar o desempenho do algoritmo com diferentes taxas de dados faltantes nos casos MCAR, exceto para a base *ArtDatasetMCAR*, onde uma situação real foi simulada, as outras duas bases foram testadas para taxas de ausência de dados de 5%, 10%, 15%, 30%, 50%, 70%, 80% e 90%. A qualidade foi medida através das métricas de *Erro Absoluto Médio* e *Erro Quadrático Médio*, conforme já explicado no Cap. 2.

A comparação será feita com outros dois algoritmos especializados em imputação do tipo MCAR, o algoritmo chamado *k-Vizinhos Mais Próximos* (*k*-VMP), similar ao algoritmo *VMP* explicado anteriormente, e o algoritmo *Decomposição em Valores Singulares Regulado*, do inglês *Regulated Singular Value Decomposition* (*rSVD*).

O *k*-VMP implementado nesse trabalho é baseado no trabalho descrito por Candillier et al. (2008) e, primeiramente, calcula a distância entre cada par de objetos, seguindo uma métrica capaz de lidar com objetos incompletos. Em seguida, para cada objeto, ele utiliza os k elementos mais próximos para estimar os valores faltantes através de uma média ponderada. Nessa implementação, foi utilizada uma combinação entre duas métricas: *Correlação de Pearson* e *Distância de Jaccard*. A correlação de *Pearson* (Stigler, 1989) mede o grau de correlação entre dois vetores de mesma dimensão:

$$\rho_{x,y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}}, \tag{5.49}$$

onde x_i e y_i, com $i = 1, ..., n$, são as duas variáveis a terem a correlação medida e

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i. \tag{5.50}$$

Os valores dessa correlação variam entre -1 e $+1$, onde $+1$ representa uma correlação positiva perfeita, -1 uma correlação negativa perfeita e 0 implica independência total entre as duas variáveis. De forma a manter o valor consistente, o cálculo desse coeficiente é efetuado apenas nos elementos onde os valores são não-nulos, em ambos os vetores.

A distância de *Jaccard*, também conhecida como índice de *Jaccard* (Jaccard, 1901), é geralmente utilizada para medir a similaridade e também a diversidade de amostras de um conjunto. Dados dois conjuntos A e B, o índice de *Jaccard* é obtido como segue:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{5.51}$$

onde $|\xi|$ é o número de elementos do conjunto ξ. Quanto maior esse índice, mais similares são os dois conjuntos. Para a situação de dados faltantes, o índice de *Jaccard* medirá a interseção e união dos conjuntos de elementos não ausentes dentro dos dois objetos. A combinação dessas duas métricas para medir a similaridade dos objetos i e j é definida como:

$$w(i, j) = (1 + \rho(i, j)) \cdot J(i, j). \tag{5.52}$$

Tendo calculado as similaridades entre cada par de objetos, basta calcular a soma ponderada dos desvios da média dos k objetos mais próximos, que contém tal atributo a ser imputado, para efetuar a predição do valor faltante, produzindo:

$$P(o, a) = \bar{r_o} + \frac{\sum_{i \in K} w(o, i) \cdot (r_{ia} - \bar{r_i})}{\sum_{i \in K} w(o, i)}, \tag{5.53}$$

onde $P(o, a)$ é a predição do valor do objeto o no atributo a, $\bar{r}_o$ representa a média dos valores atribuídos ao objeto o,r_{ia} é o valor do objeto i no atributo a, K é o conjunto de k objetos mais similares e $w(o, i)$ é a similaridade entre os objetos o e i, calculada na Eq. 5.52.

Analogamente, a predição pode ser feita na matriz transposta, onde a similaridade calculada é baseada nos atributos. Em Candillier et al. (2008), essa forma trouxe melhores resultados nos experimentos efetuados. O parâmetro otimizado para os experimentos com o k-VNP foi o valor de $k = 15$ e, ao invés de utilizar a similaridade entre os objetos, foi utilizada a similaridade entre os atributos,

proporcionando uma imputação mais rápida e de melhor qualidade.

O algoritmo *rSVD* (Funk, 2006; Paterek, 2007) é baseado na decomposição de valores singulares de uma matriz e foi aplicado ao problema de filtragem colaborativa durante o torneio Netflix (`http://www.netflixprize.com/`), em que era requisitada uma melhora de 10% em relação ao algoritmo utilizado pelo site. A base Netflix contém como objetos usuários e como atributos filmes que podem ser locados pelos usuários. O valor de cada atributo, quando existente, indica uma nota de 1 a 5 que o usuário atribuiu àquele filme. No início do torneio, o *rSVD* conquistou a posição de 3° lugar e, subsequentemente, se tornou o principal componente de outras abordagens utilizadas por outros competidores, inclusive dos dois algoritmos vencedores do torneio.

A ideia do uso da técnica de *SVD* em uma matriz esparsa é a de identificar a correlação entre o gosto de usuários parecidos em determinados filmes para gerar atributos que conseguem correlacioná-los corretamente. O termo de *regularização*, no *rSVD*, ajuda a evitar o *sobre-ajuste*, ao aprender o modelo de correlações que melhor representa os valores conhecidos da base.

Essa técnica tem características similares às de técnicas de biclusterização. Primeiramente, ela trabalha com apenas subconjuntos do espaço de busca, compactados de acordo com suas correlações e reduzindo os atributos e objetos irrelevantes para cada conjunto. Em segundo lugar, ela tende a eliminar o ruído da base por tratar e eliminar a variância encontrada durante a construção do modelo. Porém, como essa técnica define um modelo único para toda a matriz de dados, ela não permite a mesma flexibilização das técnicas de biclusterização ao definir o tamanho do subconjunto que melhor define cada região, sendo obrigada a ter um mesmo tamanho para todo o conjunto de objetos e atributos. Outra diferença é que as técnicas de biclusterização constroem modelos com interpretações diretas da relação entre objetos e atributos de cada grupo, enquanto o *rSVD* define novos atributos, tornando essa interpretação mais complicada.

Para o *rSVD*, os parâmetros foram escolhidos para gerar um modelo de 5 atributos com uma taxa de regularização de $0,015$, sendo construído em até um máximo de 200 iterações. Para o *SwarmBcluster*, foram utilizadas 5 formigas e 5 iterações e um valor de δ definido de acordo com cada base de dados. Os valores de δ foram obtidos experimentalmente, levando em conta os valores mínimo e máximo de cada base de dados. Para o caso *ArtDataset*, foi definido $\delta = 1,65$; para a base *Yeast*, o valor foi $\delta = 300$; e para a base *Jester*, o valor foi $\delta = 6$.

O primeiro experimento, sobre as bases de dados MCAR (*ArtDatasetMCAR*), MAR (*ArtDatasetMAR*) e NMAR (*ArtDatasetNMAR*), foi feito para avaliar se os valores imputados se aproximam da base de dados ruidosa *ArtDatasetRuido*, ou da base de dados real, *ArtDataset*, como os algoritmos clássicos de imputação verificados no Cap. 2. Para efeito de comparação, as Tabs. 2.6, 2.7 e 2.8 serão reproduzidas aqui através das Tabs. 5.1, 5.2 e 5.3, porém com os resultados adicionais obtidos pelos algoritmos *k*-NN, *rSVD* e *SwarmBcluster*.

Tab. 5.1: Erro absoluto médio (EAM) e erro quadrático médio (EQM) dos valores estimados por cada algoritmo junto à base de dados ArtDatasetMCAR, em relação à base original sem ruídos e à base ruidosa, com a qual foram feitos os experimentos de imputação.

ArtDatasetMCAR	**ArtDataset**		**ArtDatasetRuido**	
Algoritmo	**EAM**	**EQM**	**EAM**	**EQM**
IVM	1,32	2,04	1,42	2,99
IAT	0,78	1,06	1,09	2,02
VMP	1,90	5,77	2,07	6,72
HD	1,74	4,99	1,91	5,92
RL	0,57	0,53	0,96	1,22
MI	0,60	0,61	0,97	1,57
***k*-NN**	0,42	0,51	0,90	1,14
rSVD	0,83	0,62	1,29	1,60
SwarmBcluster	0,47	0,65	0,92	1,16

Tab. 5.2: Erro absoluto médio (EAM) e erro quadrático médio (EQM) dos valores estimados por cada algoritmo junto à base de dados ArtDatasetMAR, em relação à base original sem ruídos e à base ruidosa, com a qual foram feitos os experimentos de imputação.

ArtDatasetMAR	**ArtDataset**		**ArtDatasetRuido**	
Algoritmo	**EAM**	**EQM**	**EAM**	**EQM**
IVM	1,75	3,16	1,80	4,04
IAT	0,89	1,06	1,07	1,91
VMP	2,59	9,38	2,65	9,61
HD	2,64	8,77	2,70	9,20
RL	0,85	0,94	1,02	1,62
MI	0,87	0,96	1,05	1,71
***k*-NN**	0,35	0,49	0,88	1,16
rSVD	0,72	0,81	1,29	1,59
SwarmBcluster	0,43	0,54	0,99	1,27

A partir dessas tabelas, é possível perceber que esses três algoritmos geram imputações mais próximas da base de dados real, eliminando parte do ruído adicionado a ela. Vale notar também que os erros em relação tanto à base ruidosa quanto à base real foram melhores do que no caso das abordagens clássicas, mostrando uma maior precisão desses métodos. Porém, vale notar novamente que esse teste não tem como objetivo medir a qualidade de imputação, por conta da utilização de uma base de dados com correlações perfeitas. Além de trazer vantagens para algumas técnicas, a presença da correlação perfeita está distante de uma situação real.

Os resultados dos próximos experimentos, com a base real *Yeast*, estão reportados na Tab. 5.4.

Tab. 5.3: Erro absoluto médio (EAM) e erro quadrático médio (EQM) dos valores estimados por cada algoritmo junto à base de dados ArtDatasetNMAR, em relação à base original sem ruídos e à base ruidosa, com a qual foram feitos os experimentos de imputação.

ArtDatasetNMAR	**ArtDataset**		**ArtDatasetRuido**	
Algoritmo	**EAM**	**EQM**	**EAM**	**EQM**
IVM	1,01	1,30	1,62	2,04
IAT	0,99	1,28	1,53	1,89
VMP	1,14	1,47	1,71	2,20
HD	1,12	1,45	1,73	2,21
RL	0,95	1,22	1,55	1,98
MI	0,97	1,26	1,57	2,01
***k*-NN**	0,85	1,04	0,87	1,12
rSVD	0,92	1,21	1,29	1,53
SwarmBcluster	0,86	1,10	0,91	1,17

Nessa tabela, é importante notar primeiramente que, em até 50% de valores ausentes, as três técnicas se mantêm estáveis quanto à qualidade de suas imputações. A técnica *k*-NN não foi capaz de imputar todos os valores para o caso com 70% de valores faltantes e, por isso, teve seus resultados omitidos para evitar viés na comparação dos resultados. Porém, tanto o *SwarmBcluster* quanto o *rSVD* ainda obtiveram resultados aceitáveis, apesar de mostrarem sinais claros de piora. A partir de 80%, os algoritmos já apresentam um erro de imputação muito acima dos resultados anteriores, mostrando que, nesse ponto, a base já não contém informações suficientes para que seja possível gerar imputações coerentes. Abaixo dessa porcentagem, o *SwarmBcluster* apresenta uma média de 17% de melhora em relação ao melhor resultado dos outros dois algoritmos, conforme apresentado na última coluna da Tab. 5.4. O *rSVD* apresenta o segundo melhor resultado, porém muito acima dos erros obtidos nos resultados anteriores.

Na Tab. 5.5, são apresentados os erros quadráticos médios referentes aos mesmos experimentos da base de dados *Yeast*. Como é possível perceber, a diferença entre os resultados dos algoritmos *SwarmBcluster* e *rSVD* diminui, enquanto a diferença entre eles e o *k*-VMP aumenta, indicando uma variabilidade maior nos resultados do *k*-VMP. Novamente, a última coluna da Tab. 5.5 apresenta a porcentagem de melhora do *SwarmBcluster* sobre o melhor resultado dos dois concorrentes, para cada porcentagem de dados faltantes.

Avaliando o tempo de execução de cada algoritmo, através dos resultados da Tab. 5.6, é possível perceber que a técnica *SwarmBcluster* necessita de um tempo de processamento superior aos demais. Nos problemas com poucos valores faltantes, o *SwarmBcluster* consumiu até 6 horas para imputar todos os valores e, nos problemas acima de 50% de valores ausentes, esse tempo se estabilizou para

Tab. 5.4: Erro Absoluto Médio para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Yeast*.

% Faltantes	SwarmBcluster	k-VMP	rSVD	% de melhora
5	$19,38 \pm 17,88$	$28,62 \pm 26,36$	$22,13 \pm 18,47$	12,43
10	$18,89 \pm 17,62$	$28,67 \pm 25,13$	$22,55 \pm 20,34$	16,23
15	$18,57 \pm 17,00$	$28,40 \pm 24,34$	$22,33 \pm 19,81$	16,84
30	$19,36 \pm 18,10$	$28,67 \pm 25,04$	$23,67 \pm 21,43$	18,21
50	$20,81 \pm 20,33$	$28,86 \pm 25,41$	$26,77 \pm 24,49$	22,26
70	$25,26 \pm 29,89$	−−	$30,52 \pm 29,96$	17,23
80	$32,79 \pm 48,00$	−−	$37,55 \pm 47,73$	12,68
90	$72,83 \pm 91,36$	−−	$67,81 \pm 90,62$	−6,89

Tab. 5.5: Erro Quadrático Médio para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Yeast*.

% Faltantes	SwarmBcluster	k-VMP	rSVD	% de melhora
5	$26,37 \pm 36,54$	$38,13 \pm 53,10$	$29,68 \pm 42,30$	11,14
10	$25,83 \pm 37,96$	$38,13 \pm 53,65$	$30,37 \pm 43,65$	14,95
15	$25,18 \pm 36,05$	$37,40 \pm 50,66$	$29,85 \pm 41,34$	15,64
30	$26,50 \pm 41,50$	$38,06 \pm 54,03$	$31,93 \pm 47,29$	17,00
50	$29,10 \pm 49,88$	$38,46 \pm 53,59$	$36,28 \pm 53,17$	19,79
70	$39,14 \pm 82,84$	−−	$42,77 \pm 68,20$	8,49
80	$58,14 \pm 123,60$	−−	$60,73 \pm 119,27$	4,27
90	$116,84 \pm 181,39$	−−	$113,18 \pm 180,76$	−3,13

abaixo de 12 horas. Em comparação, as outras técnicas levaram poucos segundos para concluir a tarefa, sendo que o *rSVD* obteve uma redução de tempo conforme o número de valores faltantes aumentava, pois o custo dele é proporcional à quantidade de valores existentes na base.

A explicação para esse custo alto está justamente na construção de modelos locais feito pelo bicluster. Enquanto as outras abordagens geram um único modelo, que leva segundos para ser construído, o *SwarmBcluster* necessita construir diversos modelos para ser possível imputar toda a base de dados. De forma a mostrar essa característica, algumas estatísticas do algoritmo foram calculadas, analisando de forma aprofundada o experimento com 50% de valores faltantes. Nessa situação, o algoritmo precisou encontrar 1.094 biclusters para imputar todos os valores faltantes. Como cada bicluster encontrado, na realidade, é definido através da obtenção de 25 biclusters (5 formigas em 5 iterações), o algoritmo necessitou encontrar, de fato, um total de 27.350 biclusters. Em média, o algoritmo foi capaz de encontrar e imputar aproximadamente 38 biclusters por minuto, nesse experimento.

Tab. 5.6: Tempo computacional de cada algoritmo para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Yeast*.

% Faltantes	SwarmBcluster	k-VMP	rSVD
5	50 min.	11,7 seg.	29,9 seg.
10	1,5 hr.	19,9 seg.	29,4 seg.
15	1,8 hr.	27,4 seg.	30,4 seg.
30	5,5 hr.	46 seg.	27,4 seg.
50	12 hr.	62 seg.	11,8 seg.
70	11,5 hr.	−−	9,9 seg.
80	11 hr.	−−	7,4 seg.
90	9,5 hr.	−−	2,2 seg.

Esse conjunto de biclusters apresentou uma cobertura de 98% da base de dados, o que responde à questão da estabilização do tempo total após os 50% faltantes, uma vez que esse tempo foi suficiente para cobrir a base de dados quase totalmente. Verificando o quanto da base de dados foi imputada em certos intervalos de tempo, foi constatado também que, em 10 minutos de processamento, 50% dos valores ausentes já haviam sido imputados. Com 30 minutos esse número subiu para 70% e, em 3 horas, 90% dos valores já estavam cobertos.

Com isso, é possível perceber que um único bicluster é capaz de imputar vários valores e, inicialmente, boa parte desses valores ainda não foi imputada por outros biclusters. Com o passar das iterações, os valores imputados por um novo bicluster têm uma maior chance de já terem sido imputado por outro bicluster encontrando anteriormente, por ser permitida a sobreposição entre biclusters. Restringindo a sobreposição entre biclusters, é possível fazer com que o número de biclusters necessários para obter a cobertura total seja reduzido. Porém, isso também restringe o volume dos biclusters, já que um bicluster que foi encontrado primeiro poderá ter um volume maior, enquanto os próximos biclusters estarão restritos à área restante da base de dados.

Portanto, uma alternativa para diminuir o custo computacional é reduzir o tempo utilizado para encontrar cada bicluster. Isso pode ser feito criando um algoritmo mais eficiente ou alterando os parâmetros do *SwarmBcluster* para ser possível reduzir esse tempo, sem perda da qualidade das imputações. Uma vez que a heurística construtiva utilizada no algoritmo *SwarmBcluster* já é suficiente para encontrar um bicluster de volume aceitável, os mesmos experimentos para a base *Yeast* foram efetuados, porém utilizando apenas essa heurística construtiva. No contexto do algoritmo *SwarmBcluster*, seria equivalente a utilizar 1 formiga e 1 iteração.

Essa medida ocasiona a redução do volume médio dos biclusters, porém, conforme pode ser visto na Tab. 5.7, através das comparações do erro absoluto médio, os resultados para até 50% de valores faltantes foram ligeiramente melhores do que aqueles obtidos pelo *SwarmBcluster*. A heurística

apresenta piora apenas em porcentagens maiores, quando se torna mais difícil encontrar um bicluster restrito pela quantidade de valores faltantes.

Cabe ressaltar que os resultados obtidos pela heurística para até 50% de valores faltantes podem indicar que os biclusters encontrados pelo *SwarmBcluster*, por possuírem um volume maior, apresentam uma degradação igualmente maior, elevando assim o erro de imputação.

Tab. 5.7: Erro Absoluto Médio para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Yeast*.

% Faltantes	SwarmBcluster	Heurística
5	$19,38 \pm 17,88$	$17,73 \pm 16,33$
10	$18,89 \pm 17,62$	$17,49 \pm 16,16$
15	$18,57 \pm 17,00$	$17,32 \pm 16,13$
30	$19,36 \pm 18,10$	$18,07 \pm 17,18$
50	$20,81 \pm 20,33$	$20,03 \pm 19,89$
70	$25,26 \pm 29,89$	$31,25 \pm 39,28$
80	$32,79 \pm 48,00$	$46,77 \pm 58,22$
90	$72,83 \pm 91,36$	$95,62 \pm 102,51$

O tempo computacional, por outro lado, foi reduzido para valores mais aceitáveis, ainda que não competitivos com as outras abordagens. A Tab. 5.8 mostra os novos valores ao lado dos anteriores, onde é possível constatar uma redução de até 95% nos tempos totais, particularmente quando a porcentagem de valores faltantes é baixa.

Tab. 5.8: Tempo computacional de cada algoritmo para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Yeast*.

% Faltantes	SwarmBcluster	Heurística
5	50 min.	3 min.
10	1,5 hr.	6 min.
15	1,8 hr.	11 min.
30	5,5 hr.	55 min.
50	12 hr.	2,5 hr.
70	11,5 hr.	4 hr.
80	11 hr.	3,2 hr.
90	9,5 hr.	1,6 hr.

Como última série de experimentos, foi utilizada a base de dados *Jester*, já explicada anteriormente e com as mesmas porcentagens de valores faltantes do experimento anterior. Como essa base gera um custo computacional maior, optou-se por experimentos utilizando apenas a heurística de biclusterização, de forma a obter resultados em tempos acessíveis. Na Tab. 5.9, é possível verificar que

o *SwarmBcluster* continua obtendo valores melhores que aqueles alcançados pelos outros dois algoritmos, porém com uma porcentagem menor de melhora, uma vez que os erros médios são também relativamente menores.

Tab. 5.9: Erro Absoluto Médio para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Jester*.

% Faltantes	SwarmBcluster	k-VMP	rSVD	% de melhora
5	$3,01 \pm 2,74$	$3,21 \pm 2,70$	$3,34 \pm 2,56$	6,23
10	$3,02 \pm 2,72$	$3,24 \pm 2,71$	$3,35 \pm 2,57$	6,79
15	$3,04 \pm 2,74$	$3,24 \pm 2,73$	$3,36 \pm 2,58$	6,17
30	$3,01 \pm 2,68$	$3,27 \pm 2,74$	$3,36 \pm 2,58$	7,95
50	$3,03 \pm 2,68$	$3,33 \pm 2,77$	$3,38 \pm 2,61$	9,00
70	$3,09 \pm 2,74$	$3,42 \pm 2,83$	$3,42 \pm 2,65$	9,64
80	$3,19 \pm 2,84$	$3,46 \pm 2,87$	$3,47 \pm 2,71$	7,80
90	$3,47 \pm 3,13$	$--$	$3,66 \pm 2,92$	5,19

Vale notar que, para esse experimento, o algoritmo k-VMP obteve melhores resultados do que o *rSVD*, porém com desvio-padrão ligeiramente maior. Isso significa que, embora o erro médio para o k-VMP tenha sido menor, o *rSVD* apresentou uma discrepância menor entre os melhores e os piores erros obtidos. Isso pode ser percebido na Tab. 5.10, onde são apresentados os erros quadráticos médios. Nessa tabela, o *rSVD* apresentou um erro médio menor, na maioria dos casos, do que o k-VMP. Por sua vez, o *SwarmBcluster* manteve o menor erro, exceto no caso de 90% dos valores faltantes, onde empatou com o *rSVD*, apenas quando comparados através do erro quadrático médio.

Outro ponto a ser notado é que, por ser uma base maior, os algoritmos mantiveram estabilidade até 80% de dados faltantes e, apenas com 90%, é que começaram a apresentar uma elevação significativa no erro médio. O k-VMP, dessa vez, falhou apenas em obter todos os valores faltantes no último experimento.

Quanto ao tempo computacional, percebe-se que o tamanho da base de dados, apesar de causar influência no desempenho dos três algoritmos, causa um impacto perceptivelmente maior no *SwarmBcluster*, por este já possuir um custo superior. Comparativamente com os experimentos anteriores, tanto o *SwarmBcluster* quanto o k-NN aumentaram o tempo computacional em cerca de 15, vezes enquanto o *rSVD* aumentou seu tempo em cerca de $2,5$ vezes. Essa diferença na taxa de custo é compreensível, uma vez que os dois primeiros algoritmos aumentam o custo proporcionalmente à base de dados, enquanto o *rSVD* aumenta proporcionalmente apenas aos valores completos.

Tab. 5.10: Erro Quadrático Médio para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Jester*.

% Faltantes	SwarmBcluster	k-VMP	rSVD	% de melhora
5	$4,07 \pm 5,46$	$4,20 \pm 5,38$	$4,21 \pm 5,04$	3,09
10	$4,07 \pm 5,44$	$4,22 \pm 5,42$	$4,22 \pm 5,07$	3,55
15	$4,09 \pm 5,47$	$4,24 \pm 5,45$	$4,24 \pm 5,09$	3,54
30	$4,03 \pm 5,36$	$4,27 \pm 5,48$	$4,24 \pm 1,67$	4,95
50	$4,05 \pm 5,35$	$4,34 \pm 5,53$	$4,26 \pm 5,14$	4,93
70	$4,13 \pm 5,47$	$4,44 \pm 5,64$	$4,32 \pm 5,22$	4,40
80	$4,27 \pm 5,68$	$4,50 \pm 5,69$	$4,40 \pm 5,34$	3,04
90	$4,68 \pm 6,22$	$--$	$4,68 \pm 5,78$	0,00

Tab. 5.11: Tempo computacional de cada algoritmo para a predição de valores faltantes do tipo *MCAR* junto à base de dados *Jester*.

% Faltantes	SwarmBcluster	k-VMP	rSVD
5	3 hr.	2,4 min.	1,3 min.
10	13 hr.	3,31 min.	1,12 min.
15	43 hr.	4,47 min.	1,08 min.
30	25 hr.	7,52 min.	1,00 min.
50	42 hr.	11,00 min.	1,00 min.
70	61 hr.	14,00 min.	50 seg.
80	58 hr.	14,50 min.	50 seg.
90	70 hr.	$--$ min.	48 seg.

5.5 Caso de estudo 1: qualidade de imputação

Tendo os modelos de imputação definidos pelo conjunto de biclusters gerado pelo *SwarmBcluster* ou apenas pela heurística proposta, existem certas propriedades desses modelos que podem ser exploradas para uma melhor análise dos dados e até mesmo da qualidade da imputação.

Uma primeira propriedade é a de cada modelo possuir um indicativo da qualidade da predição. Note que, com esse indicativo, é possível saber previamente quais predições têm maiores chances de possuírem um erro menor. Isso torna-se útil, por exemplo, na aplicação de Filtragem Colaborativa, explicada no Cap. 2, uma vez que, embora os métodos tradicionais consigam gerar uma lista de predições para os valores ausentes, ainda existe a necessidade de escolher apenas um subconjunto dessa lista que melhor represente o gosto do cliente.

Esses indicativos, nos modelos gerados pelos biclusters, se apresentam através da medida de RQM, que indica o quão próximo de uma coerência perfeita o bicluster está, ou seja, o quão correta

é a predição feita por esse modelo, e através do volume, que indica a quantidade de informação que está contida nesse bicluster, de forma a reduzir o ruído interno.

Uma outra técnica de filtragem colaborativa que possui um indicativo é a *Naïve Bayes*, que retorna, além da predição, a probabilidade associada a ela. Porém, esse indicativo não leva em conta a quantidade de informação existente e utilizada para inferir tal valor. Uma predição que teve por base um ou dois objetos similares não contém informação suficiente, porém pode resultar em uma probabilidade de quase 100%, o que pode não refletir a acuidade da predição.

Para avaliar a possibilidade de utilizar os biclusters gerados como forma de avaliar a qualidade das predições, foram utilizados dois resultados dos experimentos da seção anterior: dados faltantes da base *Yeast* e da base *Jester*, com 50% de valores ausentes. Associando cada bicluster com o erro médio de predição dos valores imputados por ele, foram plotados quatro gráficos analisando RQM, volume, número de variáveis e porcentagem de valores ausentes de cada bicluster, associados ao erro médio de cada um deles.

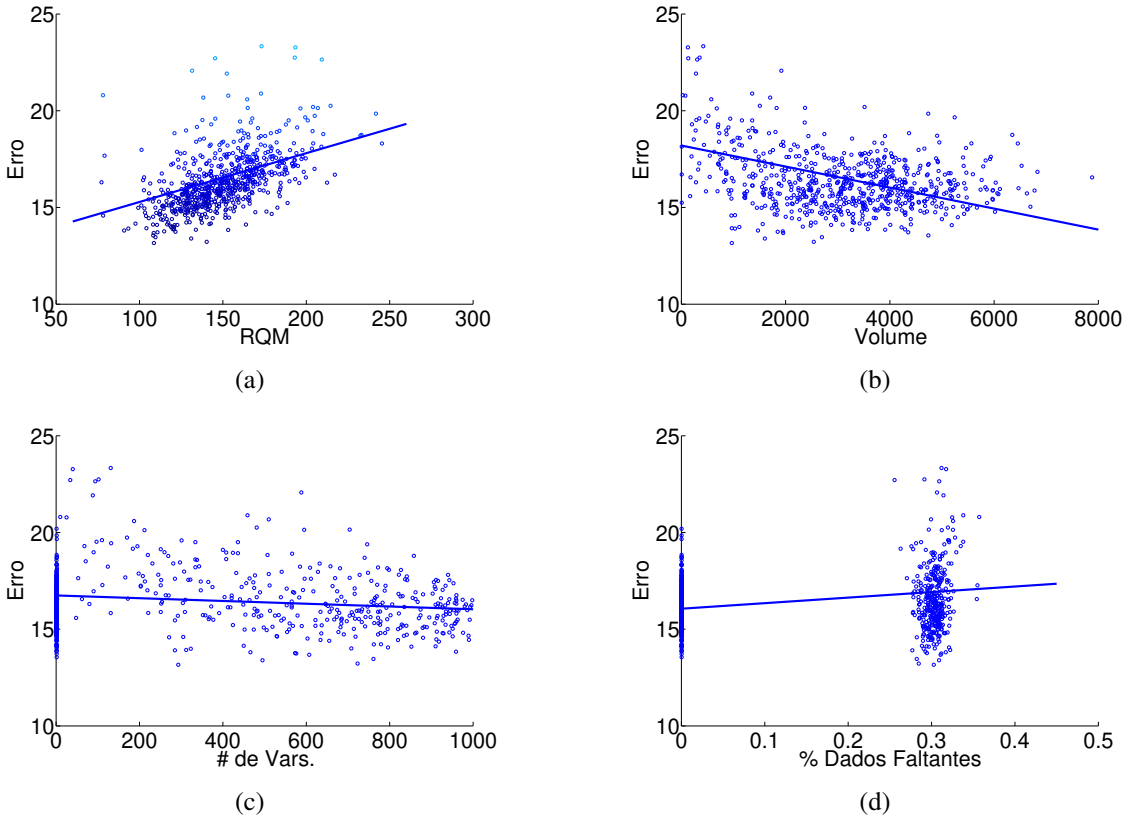


Fig. 5.3: Relação entre (a) RQM, (b) volume, (c) número de variáveis e (d) porcentagem de valores faltantes de um bicluster, em função do erro absoluto médio de imputação dele, para a base de dados *Yeast*.

Na Fig. 5.3, os gráficos representando a relação entre cada critério do bicluster e o erro de imputação são apresentados. Adicionalmente, cada gráfico contém uma linha representando a regressão linear correspondente desses pontos, para melhor visualizar a relação entre cada critério e o erro correspondente. É possível perceber que o RQM e o volume de cada bicluster apresentam uma relação mais forte na determinação da qualidade da imputação. Quanto menor o RQM menor o erro, e quanto maior o volume menor o erro, pois o bicluster contém mais informações. Já o número de variáveis detém pouca influência na qualidade da imputação e a influência da porcentagem de dados faltantes no erro não pode ser determinada, pois grande parte está concentrada em torno de 30%.

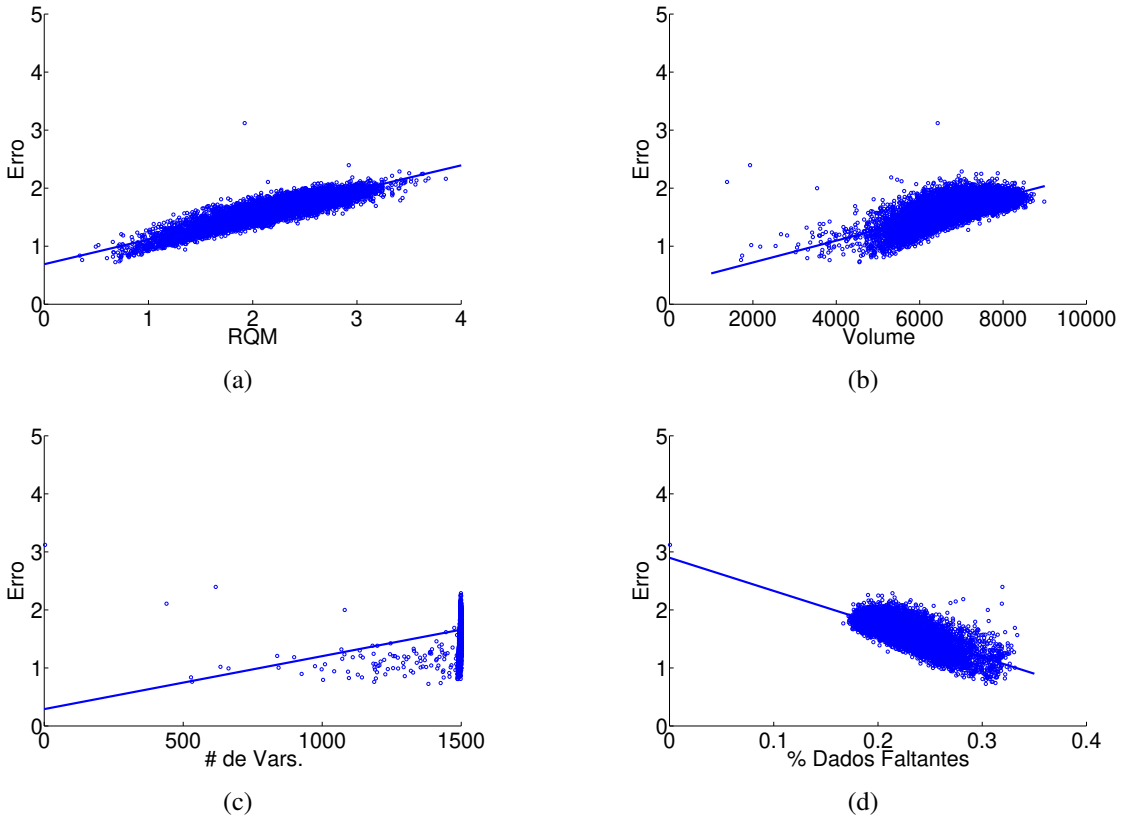


Fig. 5.4: Relação entre (a) RQM, (b) volume, (c) número de variáveis e (d) porcentagem de valores faltantes de um bicluster, em função do erro médio de imputação dele, para a base de dados *Jester*.

Para o caso da base de dados *Jester*, a Fig. 5.4 ilustra os mesmos casos anteriores. Na Fig. 5.4(a), é possível perceber que novamente o RQM tem uma relação direta com o erro de imputação: quanto menor o RQM menor o erro. Na Fig. 5.4(b), ocorre uma situação distinta, a relação do volume com o erro de imputação é um caso inverso da base de dados *Yeast*: aqui, quanto maior o volume maior o erro. Como boa parte dos biclusters continham um número de variáveis similar (em torno de 1500) a relação desse número não pôde ser mensurada adequadamente, conforme visto na Fig. 5.4(c), e, por

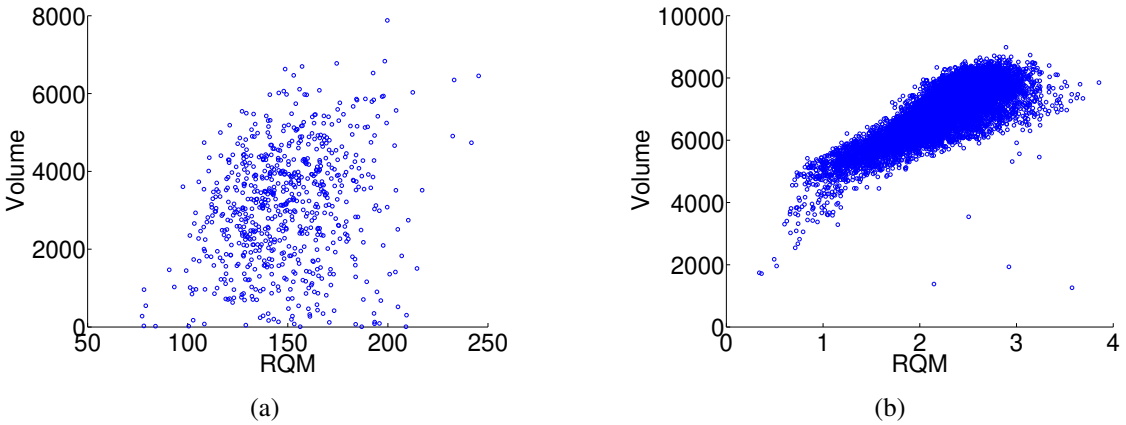


Fig. 5.5: Relação entre resíduo e volume para a base de dados (a) *Yeast* e (b) *Jester*.

conta disso, a porcentagem de valores ausentes também não tem significado, já que seria proporcional ao inverso da relação com o volume.

Para avaliar o porquê da diferente relação de volume com o erro de imputação nas duas bases, foram plotados dois gráficos traçando a relação entre resíduo e volume do bicluster. A Fig. 5.5 mostra que existe uma variabilidade grande de biclusters com combinações diferentes entre RQM e volume para a base *Yeast*, mostrando que esses dois parâmetros estão independentes na análise anterior. Já para a base *Jester*, o volume está ligado diretamente com o valor de RQM, ou seja, na análise anterior, quando era associado um maior volume com um erro maior, na verdade a relação era do RQM com o erro.

Com essa análise, é possível perceber a influência direta do RQM na qualidade da imputação e como isso pode ser utilizado para: balizar os parâmetros para gerar os biclusters; e perceber quais biclusters possuem um erro menor em relação ao conjunto total. O volume pode ser utilizado como critério de desempate, ou seja, dado que dois biclusters possuem RQM muito próximos, aquele com volume maior tende a ter um erro menor de imputação. Finalmente, a porcentagem de valores faltantes e o número de variáveis não aparentam ter influência no erro de imputação.

5.6 Caso de estudo 2: interpretabilidade dos dados

Outra propriedade interessante que pode ser extraída a partir dos modelos gerados pelos biclusters é a possibilidade de interpretar os resultados da imputação. Conforme mencionado no capítulo anterior, a biclusterização consegue delinear as relações entre os valores faltantes e os objetos e atributos pertencentes ao bicluster. Dessa forma, em uma situação em que seja necessário obter o motivo da ausência de dados, uma vez que esse é do tipo MAR ou NMAR, esses modelos trazem tais informações

intrinsecamente.

Da mesma forma, essa possibilidade de interpretação dos modelos permite que seja dada uma explicação do porquê determinado valor ter sido imputado, a partir do subconjunto de objetos e atributos ao qual ele pertence, mesmo que a ausência de dados tenha sido do tipo MCAR.

O uso dessa característica pode ser ilustrado novamente através de Filtragem Colaborativa. Dando sequência ao exemplo anterior, em que foi extraído um subconjunto da lista de recomendações através da qualidade da predição de cada bicluster, pode ser desejável que o sistema de recomendação mostre ao usuário não apenas os itens recomendados, mas também o porquê de fazerem parte dessa lista.

Exemplificando com um sistema de recomendações de filmes, o sistema pode recomendar a um usuário o filme *A* e explicar que esse filme foi recomendado pois ele também gostou dos filmes *B* e *C* e não gostou do filme *D*, todos esses contidos no mesmo bicluster utilizado para a predição. Indo um pouco além, poderia também ser feita uma análise dos filmes pertencentes ao bicluster de forma a extrair as características que eles tenham em comum, e criar então um perfil do usuário em questão.

Um exemplo de outro algoritmo, que faz essa interpretação para imputar os dados, é o SVD e suas variantes. Essa técnica, conforme dito anteriormente, compacta os atributos da base de dados de tal forma a agrupar determinadas características para os itens. Dessa forma, o algoritmo analisa as características similares de determinados itens para então inferir as recomendações. A ressalva é que os novos atributos gerados não possuem um significado claro e, portanto, não podem ser utilizados para uma interpretação explícita das recomendações. Já o bicluster informa claramente quais atributos fazem parte de determinado grupo de objetos, tornando mais clara essa interpretação.

Como experimento para verificar tal propriedade e suas implicações, foi utilizada uma base de dados denominada *Movielens* (Herlocker et al., 1999) contendo 100.000 notas dadas por 943 usuários e 1.682 filmes. A escolha dessa base é motivada pela facilidade em definir *meta-atributos* para filmes, como gêneros aos quais eles pertencem. Uma vez que essa base é incompleta, a tarefa aqui não será medir a qualidade de tais predições, mas apenas verificar a relação entre os filmes e usuários e interpretar tais resultados.

Para isso, foi extraído um único bicluster com RQM igual a $0,2$ da base de dados pré-imputada, utilizando a técnica *rSVD*. O bicluster encontrado possui 152 usuários e 303 filmes. Cada um desses filmes pode ser classificado em um ou mais gêneros dentre *Faroeste*, *Guerra*, *Thriller*, *Ficção Científica*, *Romance*, *Suspense*, *Musical*, *Terror*, *Noir*, *Fantasia*, *Drama*, *Documentário*, *Policial*, *Comédia*, *Infantil*, *Animação*, *Aventura*, *Ação* ou *Sem Classificação*. Na Fig. 5.6(a) é possível observar a distribuição de filmes em cada categoria nesse bicluster. É possível perceber, nessa figura, que esse bicluster é composto predominantemente por filmes de *drama*, seguido por filmes de *comédia*. Para um dos usuários desse bicluster, escolhido arbitrariamente, foram geradas duas listas de filmes: filmes aos quais o usuário atribuiu nota maior que 3 e filmes aos quais o usuário atribuiu nota menor ou igual

a 3. A primeira lista é referente aos filmes que o usuário gostou, e tem as distribuições dos gêneros mostradas na Fig. 5.6(b); a segunda lista remete aos filmes que o usuário não gostou, e as distribuições de gêneros estão ilustradas na Fig. 5.6(c). É possível perceber que o bicluster encontrado mostra, em sua maioria, filmes que o usuário não gostou, sendo que dos 303 filmes aqui representados, apenas 10 obtiveram nota maior que 3.

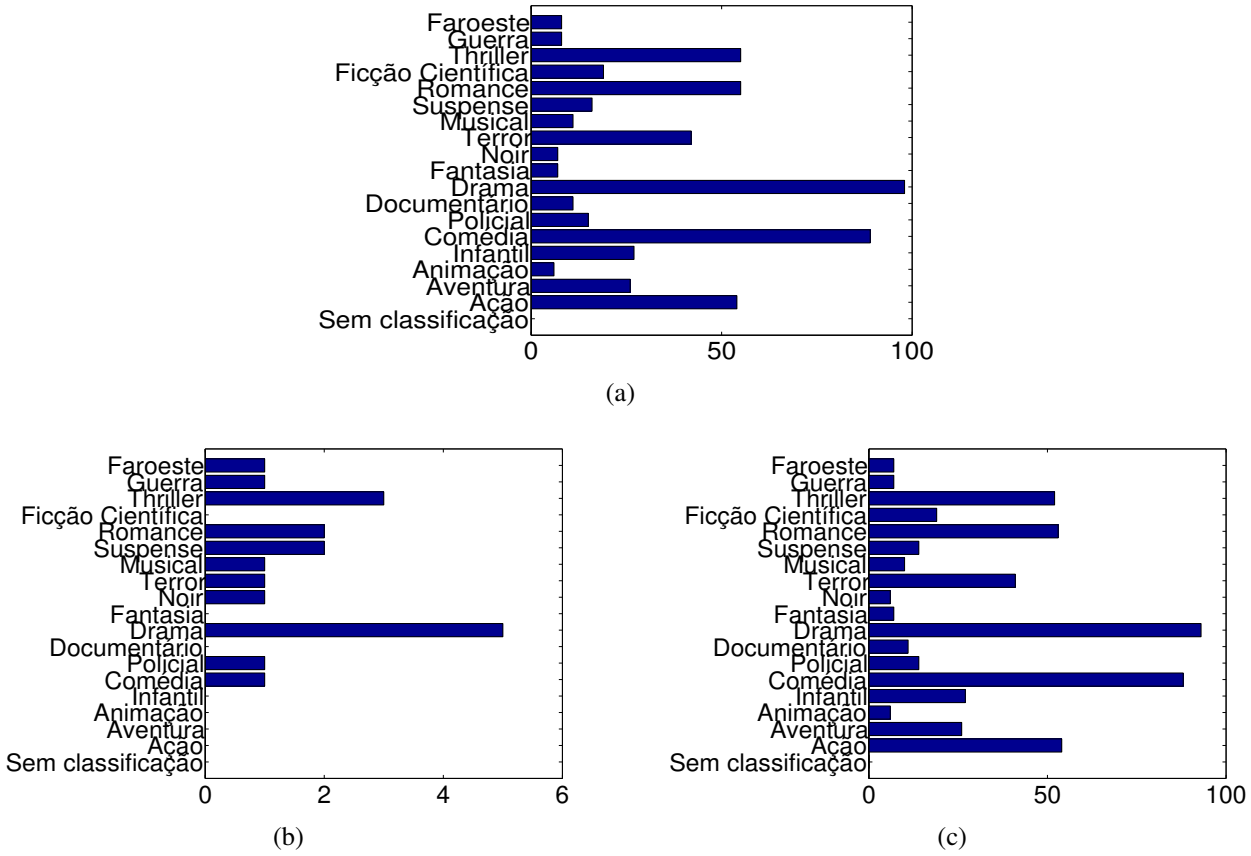


Fig. 5.6: Distribuição de gêneros de filmes do (a) bicluster completo, (b) apenas para os filmes a que o usuário deu nota positiva e (c) filmes a que o usuário deu nota negativa.

Como um filme pode pertencer a mais de uma categoria, para tentar definir o perfil de tal usuário em relação aos filmes que ele não gosta, é necessário relacionar a quais categorias conjuntas pertence a maioria dos filmes que ele gostou e que ele não gostou. Isso é ilustrado na Fig. 5.7, onde é possível observar, de forma ordenada, os tipos de filme que o usuário gosta ou não gosta.

Nessa figura, analisando os primeiros filmes, nota-se que filmes classificados como *Drama* e *Comédia* estão em destaque entre os gêneros que ele não gosta. Porém, quando o gênero *Drama* é associado com *Romance* ou *Guerra*, ele pode classificá-lo como bom. Filmes classificados como *Thriller* e *Suspense* são do gosto do usuário, enquanto *Thriller* com *Drama* e *Ação* são mal classificados,

na opinião desse usuário.

Uma vez que cada bicluster do conjunto retrata um perfil parcial do usuário, o conjunto de biclusters pode ser utilizado de diversas maneiras, quando aplicados à recomendação de itens. Um exemplo seria recomendar itens de acordo com o *humor* atual do usuário. No exemplo dos filmes, ao perceber que o usuário está procurando por filmes de *Ação*, procurar recomendações de acordo com o perfil traçado em um bicluster com predominância de filmes desse gênero. Outra utilidade é a de encontrar contra-exemplos a tipos de itens que o usuário geralmente gosta, ou seja, um usuário que costuma avaliar positivamente filmes de *Ação*, mas avalia negativamente determinados filmes desse gênero que têm participação de determinado ator, pode ter esse comportamento detectado e, assim, esses filmes serem excluídos de sua lista de recomendação automaticamente.

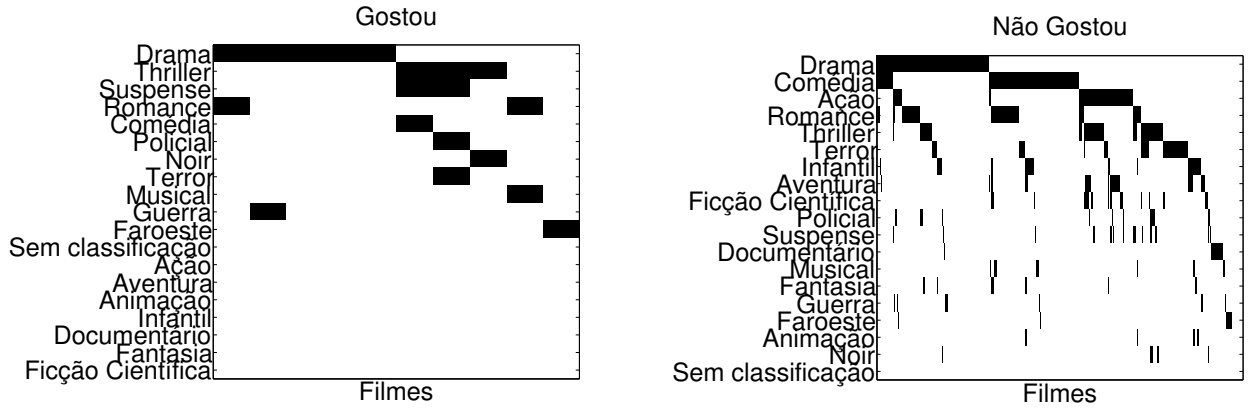


Fig. 5.7: Relação de gêneros para cada filme, dentre os filmes que o usuário (a) gostou e (b) não gostou.

Apesar do exemplo ilustrativo desse experimento, a aplicação se estende a diversas áreas em que ocorre ausência de dados, e mesmo àquelas em que os dados estejam completos e apenas é aplicado o algoritmo de biclusterização. A flexibilidade dessa técnica permite que diversas tarefas de inferência sejam cumpridas, simultaneamente e com resultados mais claros que técnicas tradicionais.

5.7 Síntese do capítulo

Este capítulo apresentou uma nova metodologia de como utilizar um conjunto de biclusters, gerado por qualquer algoritmo de biclusterização, para imputar valores ausentes de uma base de dados. Essa metodologia consiste em: primeiramente, pré-imputar os valores, utilizando uma abordagem simples e eficiente; e, após obter-se uma base de dados completa, porém ruidosa, aplicar um algoritmo de biclusterização capaz de obter cobertura máxima dos dados. Para esse fim, foi utilizado

o algoritmo *SwarmBcluster*, apresentado no Cap. 4. Finalmente, com o conjunto de biclusters encontrado, retiram-se novamente os valores previamente imputados, transformando o problema de imputação de dados em um problema de otimização quadrática, supondo que se deseja minimizar o RQM de cada bicluster. Em outras palavras, aproveita-se da medida do quão ruidoso cada bicluster é para reduzir, então, o ruído gerado durante a pré-imputação dos dados, garantindo uma imputação mais precisa e robusta.

Para obter a formulação quadrática desse problema, foi necessária a obtenção do valor das derivadas, primeira e segunda, da equação de resíduo quadrático médio para cada valor faltante no bicluster. Foi verificado que essa derivada pode assumir quatro valores distintos, dependendo da posição relativa da variável em relação ao resíduo analisado. Com a formulação definida, tornou-se necessário garantir a validade das soluções e descobrir quando o problema converge para uma solução única e factível. Isso foi feito demonstrando que a matriz hessiana do problema quadrático é inversível sempre que o bicluster apresenta uma taxa de ausência de dados, em suas linhas e colunas, menor que $\frac{1}{3}$, sendo essa uma condição suficiente independente do tamanho do bicluster. Essa condição pode ser calculada também através da dimensão do bicluster, obtendo um limitante mais preciso para a condição suficiente.

Em seguida, foram feitos testes na base de dados artificial apresentada no Cap. 2, comparando com os algoritmos apresentados naquele capítulo e dois outros algoritmos apresentados neste capítulo. Esses dois outros algoritmos apresentados foram uma variação do *k*-Vizinhos mais Próximos, que leva em conta a variabilidade dos valores de cada atributo, e um algoritmo baseado na decomposição de valores singulares (*rSVD*), que tem propriedades parecidas com a biclusterização, embora gere um modelo único para toda a base de dados.

Os experimentos iniciais serviram para mostrar que os três algoritmos, *SwarmBcluster*, *k*-VMP e *rSVD*, apresentam uma tendência em se aproximarem dos valores reais da base de dados, eliminando parte do ruído. Em seguida, foram feitos experimentos com a base *Yeast*, já utilizada anteriormente, e a base de dados *Jester*, utilizada em filtragem colaborativa. Os resultados mostraram que o *SwarmBcluster* consegue obter resultados de 5% a 20% melhores que o melhor resultado entre os outros algoritmos, enquanto mantém esses resultados com erro reduzido em bases de dados com até 70% de valores ausentes. Para uma taxa de ausência maior, ocorre uma piora considerável nos resultados, assim como nas outras abordagens.

O ponto negativo observado para essa abordagem é em relação ao custo computacional. Enquanto as outras abordagens geram um único modelo para imputar a base completa, levando de alguns segundos até poucos minutos para completar a tarefa, a biclusterização requer a construção de muitos modelos, cobrindo regiões distintas da base de dados, de tal forma que um bicluster consiga imputar pelo menos um valor ausente. Para minimizar esse custo, os parâmetros do *SwarmBcluster* foram

alterados de forma a reduzir a quantidade de modelos gerados.

Finalmente, dois casos de estudo foram conduzidos. Primeiramente, foi analisado qual critério do bicluster é responsável por uma imputação mais precisa dos valores. Nessa análise, constatou-se que o resíduo quadrático médio aparenta ter uma maior influência, seguido pelo volume do bicluster em alguns casos. O outro estudo foi para mostrar o grau de interpretabilidade dos dados gerados pelo bicluster, em que se torna possível determinar automaticamente propriedades dos objetos e atributos biclusterizados e entender o porquê de cada modelo, além de possibilitar a extração de novos conhecimentos sobre a base de dados. Em muitas aplicações, pode ser mais importante obter tal interpretação da ausência dos dados do que ter uma precisão na imputação em si.

Com este capítulo, tornam-se claras as propriedades dos métodos de biclusterização quando tratando bases de dados cujos valores apresentam incertezas. Além de obter uma qualidade melhor de imputação, os biclusters também disponibilizam informações importantes para uma pós-análise dos dados.

Capítulo 6

Conclusão

Esta tese teve o objetivo de estudar as propriedades dos métodos de biclusterização (Hartigan, 1972; Madeira & Oliveira, 2004; de França et al., 2006), mais especificamente daqueles baseados em modelos de coerência aditiva, e criar uma metodologia para se utilizar de suas propriedades na solução de problemas de tratamento de dados incertos (Longford, 2005), imputando valores ausentes e reduzindo o ruído da base de dados.

O Cap. 2 introduziu o problema da incerteza, contida em bases de dados presentes em diversas situações práticas, e ilustrou situações comuns em que isso ocorre. Em seguida, foram apresentadas definições formais para as três principais causas da ausência de valores, de acordo com a dependência entre valores de diferentes atributos: independente de qualquer evento (MCAR), dependente de valores de outros atributos (MAR) e dependente do próprio valor do atributo (NMAR). Alguns métodos para solucionar tais problemas, com diferentes funcionalidades e características, foram apresentados juntamente com experimentos aplicados a uma base de dados artificial, com o propósito de apresentar de forma didática o funcionamento dessas soluções e ilustrar os desafios desse problema. Por fim, foi também apresentada uma aplicação prática muito estudada atualmente, que envolve diretamente o problema de imputação de dados, conhecida como Filtragem Colaborativa (Candillier et al., 2008).

O conceito de biclusterização, uma forma de extrair conhecimentos de uma base de dados seguindo um modelo local, foi formalizado no Cap. 3, inicialmente de forma generalizada. O modelo local definido pelo bicluster leva em conta apenas parte dos objetos e dos atributos dos dados, de forma que o significado que se deseja obter seja evidenciado. As definições mais comuns para um bicluster são aquelas com todos os seus valores constantes, linhas com valores constantes, colunas com valores constantes ou apresentando valores (ou tendências) coerentes. Além disso, pode-se impor que esses biclusters obedeçam perfeitamente à propriedade requisitada ou pode-se relaxar essa restrição, tentando minimizar o erro entre o bicluster encontrado e um modelo ideal para esse bicluster. Novamente, para um melhor entendimento, foram apresentadas diversas técnicas de biclusterização para

cada um dos tipos frequentemente estudados e, em seguida, focou-se nos biclusters com coerência aditiva.

Aprofundando o estudo de modelos de coerência aditiva, o Cap. 4 introduziu uma nova heurística para a busca de um bicluster, que minimizasse o erro residual, seguindo esse modelo, e maximizasse seu tamanho. Para buscar um conjunto desses biclusters, de tal forma que esse conjunto maximizasse a cobertura da base de dados, foi criado o algoritmo *SwarmBcluster*. Este algoritmo é baseado nos conceitos de inteligência de enxame (Dorigo, 1992), sendo um algoritmo capaz de obter um conjunto de biclusters que otimiza cada um dos critérios, atingindo seu objetivo. Isso foi verificado através de resultados experimentais, quando o *SwarmBcluster* foi comparado com outras abordagens da literatura. Alguns testes complementares foram efetuados para mostrar características interessantes dos biclusters encontrados, pertinentes aos problemas de base de dados com valores incertos. O primeiro experimento foi referente aos dados ruidosos e como os modelos, definidos por cada bicluster, tendem a minimizar tal ruído. O segundo experimento mostrou que as técnicas de biclusterização conseguem capturar a correlação parcial entre atributos, permitindo a descoberta de boa parte daqueles atributos que são responsáveis pela ausência de determinados valores, quando do tipo MAR, e a faixa de valores de certo atributo que aumenta as chances de elementos desse atributo estarem ausentes, para os casos NMAR.

Uma vez estabelecidos os conceitos do problema de valores ausentes e da tarefa de biclusterização, além da criação de um algoritmo capaz de encontrar um conjunto de biclusters com características desejáveis, foi então definida, no Cap. 5, uma metodologia capaz de encontrar um conjunto de biclusters, em uma base de dados incompleta, e utilizar-se dos modelos definidos por eles para imputar os valores ausentes. O primeiro passo foi encontrar um bicluster dentro da base de dados, tendo em vista a ausência de valores. Conforme experimento no capítulo anterior, percebeu-se que as técnicas de biclusterização são capazes de encontrar bons modelos, mesmo na presença de ruído. Portanto, a solução proposta foi pré-imputar os valores, com alguma técnica simples e eficiente, transformando uma base incompleta em uma base ruidosa, e, em seguida, utilizar essa base para definir o conjunto de biclusters. Tendo os modelos locais definidos pelo conjunto de biclusters, os valores pré-imputados podem então ser retirados novamente da base, para em seguida imputá-los novamente, agora empregando os modelos livres de ruído.

Essa imputação final é feita de forma que os valores minimizem o ruído do modelo de coerência definido por cada bicluster. Esse problema de minimização foi então transformado em um problema de otimização quadrática (Goldfarb & Idnani, 1983), permitindo o uso de técnicas já consolidadas que obtêm soluções exatas de forma eficiente. Toda a formulação matemática do problema foi apresentada, além do cálculo de derivadas da função residual do bicluster, necessárias para viabilizar a obtenção da solução. Em seguida, foram estudadas as condições suficientes, nas propriedades do

bicluster, de forma a garantir a factibilidade da solução para o problema quadrático. Finalmente, com a consolidação do algoritmo de imputação de dados utilizando biclusterização, experimentos foram conduzidos mostrando que o erro de estimação, com o uso de tal metodologia, é menor do que aqueles obtidos com outros métodos consolidados na área. Como contrapartida da qualidade dos resultados, o tempo computacional utilizado por essa metodologia é superior ao demandado pelas demais abordagens, o que pode ser explicado pelo fato de que o método de biclusterização necessita definir muitos modelos locais de uma base de dados, enquanto as outras abordagens geram um modelo único para toda a base.

Duas propriedades das soluções geradas pelo método de biclusterização também foram estudadas. A primeira mostra que é possível ter um indicativo da qualidade da imputação através das próprias características de cada bicluster, como o erro residual e o seu tamanho. Isso pode ser útil em diversas aplicações em que as predições mais precisas devem ser apresentadas primeiro, ou então para medir novamente valores que obtiveram uma predição mais imprecisa. A segunda propriedade estudada é a facilidade em interpretar os modelos. Uma vez que o resultado do bicluster apresenta diretamente o subconjunto de objetos e atributos utilizados para a construção do modelo, é fácil determinar o motivo de certas predições, permitindo gerar um modelo qualitativo de classificação de objetos e/ou atributos, ou então inferir outras características tanto da base de dados quanto da ausência dos dados.

6.1 Discussão

Apesar de responder diversas questões e contribuir com resultados positivos tanto na área de análise de dados na presença de incerteza quanto na área de biclusterização, existem ainda algumas questões em aberto e pontos a serem avaliados e melhorados.

Um dos pontos em aberto diz respeito à parametrização dos algoritmos introduzidos neste trabalho. Apesar de todos os parâmetros terem sido obtidos experimentalmente, é necessário realizar uma análise da sensibilidade e influência de cada parâmetro na qualidade dos resultados, facilitando a escolha destes ao aplicar o algoritmo em uma nova base de dados. Além disso, é importante notar que certos pontos dos algoritmos permitem diversas escolhas não exploradas aqui, como, por exemplo, a utilização de uma informação heurística para auxiliar a construção de uma solução durante os passos do ACO.

Vale ressaltar que a análise de sensibilidade dos parâmetros deve ser feita tanto na aplicação para biclusterização como na tarefa de imputação de valores faltantes. Um dos parâmetros do algoritmo de biclusterização, o δ, é de considerável importância para a qualidade final da imputação, conforme visto neste trabalho. Esse parâmetro serve como limitante inferior da qualidade do bicluster para que este possa ser usado nas diversas tarefas. Um estudo aprofundado desse parâmetro e uma forma de

estimar seu valor ótimo, de acordo com as propriedades da base de dados, pode ser de grande valia para facilitar o procedimento de imputação.

Outro estudo que deve ser feito é a adaptação de outras técnicas de biclusterização para ser possível obter máxima cobertura e comparar o desempenho dos biclusters gerados por elas na imputação de dados.

A otimização do desempenho computacional do algoritmo de imputação também deve ser estudada. Uma das formas, conforme citado anteriormente, é alterando o algoritmo de modo que seja possível obter biclusters com volumes maiores e sobreposições menores, para que seja possível a obtenção de menos biclusters visando obter total cobertura da base de dados.

Outra forma possível para reduzir o custo computacional é através da paralelização desse processo. É possível perceber que a obtenção de cada bicluster ocorre de forma quase independente, podendo cada um deles ser obtido utilizando um processador diferente e independente. Isso é possível através de programação maciçamente paralela via *GPU* (Buck, 2007; Harish & Narayanan, 2007) onde são utilizados os vários processadores independentes dentro de uma placa de vídeo para computar diversos processos em paralelo.

6.2 Perspectivas Futuras

Uma vez que o tema biclusterização é, de certa forma, recente, existem muitos aspectos ainda a serem explorados e desafios a serem vencidos. Fica em aberto a perspectiva de novas pesquisas e expansões dos conceitos introduzidos aqui, tanto na imputação de valores faltantes, utilizando biclusters, como na utilização da técnica de biclusterização em outras áreas de mineração de dados.

Até então, as técnicas de biclusterização têm sido utilizadas, em sua grande maioria, para problemas de análise de expressões gênicas na área de biologia (Agrawal et al., 1998; Cheng & Church, 2000). Porém, essa técnica apresenta um grande potencial para as áreas de análise estatística, definições de modelos, inferência de conhecimento, classificação de dados, dentre outras, justamente por sua característica de encontrar padrões multi-facetados através de modelos locais.

Uma aplicação direta dos resultados dessa tese é o estudo de sistemas de recomendação e filtragem colaborativa, descritos anteriormente. Com o uso da biclusterização, é possível não só imputar os dados faltantes para estimar as notas que clientes poderiam dar a certos produtos, como também adquirir diversas informações de preferências do cliente.

Outra aplicação possível para a biclusterização é a mineração de textos, que tem como objetivo encontrar termos, ou sequências deles, que identifiquem os possíveis assuntos de um determinado texto. A técnica de biclusterização se torna interessante nesse cenário, pois um texto pode pertencer a diversos assuntos e cada texto contém apenas um sub-conjunto de termos, dentre o universo de

termos existente. Apesar de algumas tentativas terem sido feitas para essa aplicação (Dhillon, 2001; Feldman & Sanger, 2006; de Castro et al., 2007b; de Castro et al., 2010), um estudo aprofundado e, possivelmente a criação de um novo algoritmo de biclusterização pode se tornar necessário, uma vez que, embora esse problema trate de uma matriz esparsa, o objetivo aqui nem sempre é a imputação dos dados.

Também de interesse prático, a mineração de padrões, que tem como objetivo encontrar regras de associações geralmente referente a comércios ou negócios, pode se beneficiar das propriedades da biclusterização. Como as bases de dados de mineração de padrões são geralmente representadas por diversas amostras (objetos) contendo eventos ou padrões (atributos), é desejável encontrar subconjuntos dessas amostras com subconjuntos desses padrões que indiquem um padrão comum a ser explorado.

Como último exemplo de possível aplicação de técnicas de biclusterização, pode ser citada a predição de séries temporais. Similarmente à mineração de padrões, nas séries temporais pode ser necessário perceber padrões que ocorreram apenas durante um determinado período de tempo e em parte das amostras, de tal forma que, caso esse evento venha a ocorrer novamente, o sistema esteja preparado para reagir a essa mesma situação.

Conforme é possível perceber, a área de biclusterização, embora já tenha sido formalizada e possua diversas técnicas específicas para as suas várias definições, ainda precisa de um amplo estudo e, possivelmente, novas definições de correlação, métricas de qualidade e técnicas para se tornar uma ferramenta capaz de resolver os diversos problemas propostos em mineração de dados, incluindo os desafios associados à escalabilidade e distributividade das bases de dados a serem analisadas.

Referências Bibliográficas

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749.

Agrawal, R., Gehrke, J., Gunopulus, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. Em *Proc. of the ACM/SIGMOD Int. Conference on Management of Data*, (pp. 94–105).

Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., Powell, J. I., Yang, L., Marti, G. E., Moore, T., Hudson Jr., J., Lu, L., Lewis, D. B., Tibshirani, R., Sherlock, G., Chan, W. C., Greiner, T. C., Weisenburger, D. D., Armitage, J. O., Warnke, R., Levy, R., Wilson, W., Grever, M. R., Byrd, J. C., Botstein, D., Brown, P. O., & Staudt, L. M. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, *403*, 503–510.

Applegate, D. (2006). *The traveling salesman problem: a computational study*. Princeton Univ Pr.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, *53*, 370–418.

Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping Multidimensional Data*, (pp. 25–71).

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is nearest neighbor meaningful? Em *In Int. Conf. on Database Theory*, (pp. 217–235). Springer.

Bezdek, J., Ehrlich, R., et al. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, *10*(2-3), 191–203.

Billen, J. (2006). Signal variety and communication in social insects. Em *Proceedings of the Section Experimental and Applied Entomology-Netherlands Entomological Society*, vol. 17, (p. 9).

Bishop, J. (1989). Stochastic searching networks. Em *Memorias del First IEEE Conference on Artificial Neural Networks*, (pp. 329–331).

Blatt, M., Wiseman, S., & Domany, E. (1996). Superparamagnetic clustering of data. *Physical Review Letters*, *76*(18), 3251–3254.

Blum, C., & Dorigo, M. (2004). The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *34*(2), 1161–1172.

Buck, I. (2007). GPU computing with nvidia cuda. Em *ACM SIGGRAPH 2007 courses*, (p. 6). ACM.

Califano, A., Stolovitzky, G., Tu, Y., et al. (2000). Analysis of gene expression microarrays for phenotype classification. Em *Proc Int Conf Intell Syst Mol Biol*, vol. 8, (pp. 75–85).

Candillier, L., Jack, K., Fessant, F., & Meyer, F. (2008). *State-of-the-Art Recommender Systems*, cap. 1, (pp. 1–22). Information Science Publishing.

Chatterjee, S., & Hadi, A. (1986). Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, *1*(3), 379–393.

Cheng, Y., & Church, G. M. (2000). Biclustering of expression data. Em *Proc. of the 8th Int. Conf. on Intelligent Systems for Molecular Biology*, (pp. 93–103).

Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., & Davis, R. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, *2*, 65–73.

Coelho, G. P., de França, F. O., & Von Zuben, F. J. (2008). A Multi-Objective Multipopulation Approach for Biclustering. Em P. J. Bentley, D. Lee, & S. Jung (Eds.) *Artificial Immune Systems, Proc. of the 7th International Conference on Artificial Immune Systems (ICARIS)*, vol. 5132 de *Lecture Notes in Computer Science*, (pp. 71–82).

Coelho, G. P., de França, F. O., & Von Zuben, F. J. (2009a). Improving a multi-objective multipopulation artificial immune network for biclustering. Em *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, (pp. 2748–2755). Institute of Electrical and Electronics Engineers Inc., The.

Coelho, G. P., de França, F. O., & Von Zuben, F. J. (2009b). Multi-objective biclustering: When non-dominated solutions are not enough. *Journal of Mathematical Modelling and Algorithms*, *8*(2), 175–202.

Cohen, J. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences*. Lawrence Erlbaum.

Colantonio, A., Di Pietro, R., Ocello, A., & Verde, N. (2010). ABBA: Adaptive bicluster-based approach to impute missing values in binary matrices. Em *Proceedings of the 2010 ACM Symposium on Applied Computing*, (pp. 1026–1033). ACM.

Courtois, P., & Heymans, F. (1991). A simulation of the construction process of a termite nest. *Journal of Theoretical Biology*, *153*(4), 469–475.

de Castro, L., & Von Zuben, F. (2001). aiNet: An artificial immune network for data analysis. *Data Mining: a Heuristic approach*, *1*, 231–259.

de Castro, L. N., & Von Zuben, F. J. (2000). An evolutionary immune network for data clustering. Em *Proceedings of 6th Brazilian Symposium on Neural Networks (SBRN 2000*, (pp. 84–89). IEEE Computer Society.

de Castro, P. A. D., de França, F. O., Ferreira, H. M., Coelho, G. P., & Von Zuben, F. J. (2010). Query expansion using an immune-inspired biclustering algorithm. *Natural Computing*, (pp. 1–24).

de Castro, P. A. D., de França, F. O., Ferreira, H. M., & Von Zuben, F. J. (2007a). Applying Biclustering to Perform Collaborative Filtering. Em *Proc. of the 7th International Conference on Intelligent Systems Design and Applications*, (pp. 421–426). Rio de Janeiro, Brazil.

de Castro, P. A. D., de França, F. O., Ferreira, H. M., & Von Zuben, F. J. (2007b). Applying Biclustering to Text Mining: An Immune-Inspired Approach. Em L. N. de Castro, F. J. Von Zuben, & H. Knidel (Eds.) *Artificial Immune Systems, Proc. of the 6th International Conference on Artificial Immune Systems (ICARIS)*, vol. 4628 de *Lecture Notes in Computer Science*, (pp. 83–94). Santos, Brazil.

de Castro, P. A. D., de França, F. O., Ferreira, H. M., & Von Zuben, F. J. (2007c). Evaluating the Performance of a Biclustering Algorithm Applied to Collaborative Filtering: A Comparative Analysis. Em *Proc. of the 7th International Conference on Hybrid Intelligent Systems*, (pp. 65–70). Kaiserslautern, Germany.

de França, F. O., Bezerra, G. P., & Von Zuben, F. J. (2006). New Perspectives for the Biclustering Problem. Em *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, (pp. 753–760). IEEE.

de França, F. O., Coelho, G. P., & Von Zuben, F. J. (2008). bicACO: An Ant Colony Inspired Biclustering Algorithm. Em *6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008. Proceedings*, vol. 5217, (pp. 401–402). Springer.

de França, F. O., & Von Zuben, F. J. (2010). Finding a high coverage set of δ-biclusters with swarm intelligence. Em *Evolutionary Computation (CEC), 2010 IEEE Congress on*, (pp. 1–8). Barcelona, Spain.

de França, F. O., Coelho, G. P., Castro, P. A. D., & Von Zuben, F. J. (2010). Conceptual and Practical Aspects of the aiNet Family of Algorithms. *International Journal of Natural Computing Research*, *1*(1), 1–35.

de França, F. O., Coelho, G. P., & Von Zuben, F. J. (2009). Coherent recommendations using biclustering. Em *Proc. of the XXX Congresso Ibero-Latino-Americano de Métodos Computacionais em Engenharia (CILAMCE)*, (pp. 1–15). Armação de Búzios, RJ, Brazil.

de França, F. O., Von Zuben, F. J., & de Castro, L. N. (2004a). A max min ant system applied to the capacitated clustering problem. Em *Machine Learning for Signal Processing, 2004. Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*, (pp. 755–764).

de França, F. O., Von Zuben, F. J., & de Castro, L. N. (2004b). Definition of Capacited p-Medians by a Modified Max Min Ant System with Local Search. Em *Neural Information Processing*, (pp. 1094–1100). Springer.

de França, F. O., Von Zuben, F. J., & de Castro, L. N. (2005). Max min ant system and capacitated p-medians: Extensions and improved solutions. *Informatica*, *29*, 163–171.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. Em *Proc. of the 7th Int. Con. on Knowledge Discovery and Data Mining*, (pp. 269–274).

Divina, F., & Aguilar–Ruiz, J. S. (2007). A Multi-Objective Approach to Discover Biclusters in Microarray Data. Em *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'07)*, (pp. 385–392). London, UK.

Divina, F., et al. (2006). Biclustering of expression data with evolutionary computation. *IEEE transactions on knowledge and data engineering*, (pp. 590–602).

Donders, A., van der Heijden, G., Stijnen, T., & Moons, K. (2006). Review: a gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, *59*(10), 1087–1091.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. Tese de doutorado, Politecnico di Milano, Italy.

Feldman, R., & Sanger, J. (2006). *The Text Mining Handbook*. Cambridge University Press.

Feo, T., & Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, *8*(2), 67–71.

Ford, B. L. (1983). An overview of hot-deck procedures. Em *Incomplete data in sample surveys*, vol. 3.

Funk, S. (2006). Netflix update: Try this at home. http://sifter.org/ simon/journal/20061211.html.

Getz, G., Levine, E., & Domany, E. (2000). Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences*, *97*(22), 12079–12084.

Gibbons, J., & Chakraborti, S. (2003). *Nonparametric statistical inference*. M. Dekker.

Giráldez, R., Divina, F., Pontes, B., & Aguilar–Ruiz, J. S. (2007). Evolutionary Search of Biclusters by Minimal Intrafluctuation. Em *Proceedings of the IEEE International Fuzzy Systems Conference (FUZZ–IEEE 2007)*, (pp. 1–6). London, UK.

Glover, F., & Kochenberger, G. (2003). *Handbook of metaheuristics*. Springer.

Glover, F., & Marti, R. (2006). Tabu search. *Metaheuristic Procedures for Training Neural Networks*, (pp. 53–69).

Goldberg, D. (1989). *Genetic Algorithms in Search and Optimization*. Addison-wesley.

Goldfarb, D., & Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, *27*, 1–33.

Goss, S., Aron, S., Deneubourg, J., & Pasteels, J. (1989). Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, *76*(12), 579–581.

Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual Review of Psychology*, *60*(1), 549–576.

Grzymala-Busse, J. W., & Hu, M. (2001). A comparison of several approaches to missing attribute values in data mining. Em *RSCTC '00: Revised Papers from the Second International Conference on Rough Sets and Current Trends in Computing*, (pp. 378–385). London, UK: Springer-Verlag.

Han, J., & Kamber, M. (2006). *Data mining: concepts and techniques*. Morgan Kaufmann.

Harish, P., & Narayanan, P. (2007). Accelerating large graph algorithms on the GPU using CUDA. *High Performance Computing–HiPC 2007*, (pp. 197–208).

Hartigan, J. (1972). Direct clustering of a data matrix. *Journal of the American Statistical Association*, *67*(337), 123–129.

Hartwig, F., & Dearing, B. (1979). *Exploratory data analysis*. Sage Publications, Inc.

Harville, D. (2001). *Matrix algebra: exercises and solutions*, cap. 14, (pp. 99–100). Springer Verlag.

Harville, D. (2008). *Matrix algebra from a statistician's perspective*, cap. 14, (p. 247). Springer-Verlag New York Inc.

Herlocker, J., Konstan, J., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. Em *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (p. 237). ACM.

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 53.

Hochberg, Y., & Tamhane, A. (1987). *Multiple comparison procedures*, vol. 82. Wiley Online Library.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, *37*, 547–579.

Jackson, D., Holcombe, M., & Ratnieks, F. (2004). Trail geometry gives polarity to ant foraging networks. *Nature*, *432*(7019), 907–909.

Jain, A., Murty, M., & Flynn, P. (1999). Data clustering: a review. *ACM Computing Surveys (CSUR)*, *31*(3), 264–323.

Johnson, S. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Em *IEEE International Conference on Neural Networks, 1995. Proceedings.*, vol. 4.

Kluger, Y., Basri, R., Chang, J., & Gerstein, M. (2003). Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, *13*(4), 703.

Kruskal Jr, J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, *7*(1), 48–50.

Lazzeroni, L., & Owen, A. (2002). Plaid models for gene expression data. *Statistica Sinica*, *12*(1), 61–86.

Lemire, D. (2005). Scale and translation invariant collaborative filtering systems. *Information Retrieval*, *8*(1), 129–150.

Lewis, D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *Machine Learning: ECML-98*, (pp. 4–15).

Little, R. J. A., & Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. Wiley.

Longford, N. (2005). *Missing Data and Small-Area Estimation*. Springer Science+ Business Media, Inc.

Madeira, S. C., & Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, *1*(1), 24–45.

Maulik, U., Mukhopadhyay, A., Bandyopadhyay, S., Zhang, M. Q., & Zhang, X. (2008). Multiobjective fuzzy biclustering in microarray data: Method and a new performance measure. Em *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, (pp. 1536–1543). Hong Kong, China.

McKnight, P. E., McKnight, K. M., Sidani, S., & Figueredo, A. J. (2007). *Missing Data: A Gentle Introduction*. The Guilford Press.

Mirkin, B. (1996). *Mathematical Classification and Clustering*. Springer.

Mitra, S., & Banka, H. (2006). Multi-objective evolutionary biclustering of gene expression data. *Pattern Recognition*, *39*, 2464–2477.

Mitra, S., Banka, H., & Pal, S. K. (2006). A MOE framework for Biclustering of Microarray Data. Em *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, (pp. 1154 – 1157). Hong Kong, China.

Molenberghs, G., Beunckens, C., Sotto, C., & Kenward, M. (2008). Every missing not at random model has got a missing at random counterpart with equal fit. *Journal of the Royal Statistical Society, Series B*, *70*, 371–388.

Murali, T., & Kasif, S. (2002). Extracting conserved gene expression motifs from gene expression data. Em *Pacific Symposium on Biocomputing 2003: Kauai, Hawaii, 3-7 January 2003*, (p. 77). World Scientific Pub Co Inc.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. Em *Proceedings of KDD Cup and Workshop*. Citeseer.

Peeters, R. (2003). The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, *131*(3), 651–654.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm–a novel tool for complex optimisation problems. Em *Memorias del Innovative Production Machines and Systems Virtual Conference*.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences*, *179*(13), 2232–2248.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. New York: John Wiley & Sons.

Sande, I. G. (1983). Hot-deck imputation procedures. Em W. Madow, & I. Olkin (Eds.) *Incomplete data in sample surveys*, vol. 3, (pp. 339–349). New York: Academic Press.

Schafer, J., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. *The Adaptive Web*, (pp. 291–324).

Scheffer, J. (2002). Dealing with missing data. Em *Research Letters in the Information and Mathematical Sciences*, (pp. 153–160).

Shah-Hosseini, H. (2007). Problem solving by intelligent water drops. Em *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, (pp. 3226–3231).

Steck, H. (2010). Training and testing of recommender systems on data missing not at random. Em *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 713–722). ACM.

Stigler, S. (1989). Francis Galton's account of the invention of correlation. *Statistical Science*, *4*(2), 73–79.

Stützle, T., & Dorigo, M. (1999). ACO algorithms for the quadratic assignment problem. *New ideas in optimization*, (pp. 33–50).

Stützle, T., & Hoos, H. H. (2000). MAX-MIN ant system. *Future Generation Computer Systems*, *16*(8), 889–914.

Symeonidis, P., Nanopoulos, A., Papadopoulos, A., & Manolopoulos, Y. (2007). Nearest-biclusters collaborative filtering with constant values. Em *Advances in Web Mining and Web Usage Analysis*, vol. 4811 de *Lecture Notes in Computer Science*, (pp. 36–55). Philadelphia, USA: Springer-Verlag.

Tanay, A., Sharan, R., & Shamir, R. (2002). Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, *18*(Suppl 1), S136.

Tang, C., Zhang, L., Zhang, I., & Ramanathan, M. (2001). Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. Em *Proc. of the 2nd IEEE Int. Symposium on Bioinformatics and Bioengineering*, (pp. 41–48).

Tavazoie, S., Hughes, J., Campbell, M., Cho, R., & Church, G. (1999). Systematic determination of genetic network architecture. *Nature Genetics*, *22*, 281–285.

Yu, K., Schwaighofer, A., Tresp, V., Xu, X., & Kriegel, H.-P. (2004). Probabilistic memory-based collaborative filtering. *IEEE Trans. on Knowledge and Data Engineering*, *16*(1), 56–69.

Trabalhos Publicados Pelo Autor

1. de França, F. O., & Von Zuben, F. J. (2010). Finding a high coverage set of δ-biclusters with swarm intelligence. In *Proceedings of the 12th IEEE Congress on Evolutionary Computation*, (pp. 1–8). Institute of Electrical and Electronics Engineers Inc., The.

2. de França, F. O., Coelho, G. P., & Von Zuben, F. J. (2010). On the diversity mechanisms of opt-aiNet: A comparative study with fitness sharing. In *Proceedings of the 12th IEEE Congress on Evolutionary Computation*, (pp. 1–8). Institute of Electrical and Electronics Engineers Inc., The.

3. de França, F. O., Coelho, G. P., Castro, P. A. D., & Von Zuben, F. J. (2010). Conceptual and practical aspects of the aiNet family of algorithms. *International Journal of Natural Computing Research*, *1*(1), 1–35.

4. de Castro, P. A. D., de França, F. O., Ferreira, H. M., Coelho, G. P., & Von Zuben, F. J. (2010). Query expansion using an immune-inspired biclustering algorithm. *Natural Computing*, (pp. 1–24).

5. de França, F. O., Coelho, G. P., & Von Zuben, F. J. (2009). Coherent recommendations using biclustering. In *Proc. of the XXX Congresso Ibero-Latino-Americano de Métodos Computacionais em Engenharia (CILAMCE)*, (pp. 1–15). Armação de Búzios, RJ, Brazil.

6. Coelho, G. P., de França, F. O., & Von Zuben, F. J. (2009). Multi-objective biclustering: When non-dominated solutions are not enough. *Journal of Mathematical Modelling and Algorithms*, *8*(2), 175–202.

7. Coelho, G. P., de França, F. O., & Von Zuben, F. J. (2009). Improving a multi-objective multipopulation artificial immune network for biclustering. In *Proceedings of the Eleventh Congress on Evolutionary Computation*, (pp. 2748–2755). The Institute of Electrical and Electronics Engineers.

8. Coelho, G. P., de França, F. O., & Von Zuben, F. J. (2008). A Multi-Objective Multipopulation Approach for Biclustering. In P. J. Bentley, D. Lee, & S. Jung (Eds.) *Artificial Immune Systems, Proc. of the 7th International Conference on Artificial Immune Systems (ICARIS)*, vol. 5132 of *Lecture Notes in Computer Science*, (pp. 71–82).

9. de França, F. O., Coelho, G. P., & Von Zuben, F. J.(2008). bicACO: An Ant Colony Inspired Biclustering Algorithm. In *6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008. Proceedings*, vol. 5217, (pp. 401–402). Springer.

10. de Castro, P. A. D., de França, F. O., Ferreira, H. M., & Von Zuben, F. J. (2007). Applying Biclustering to Perform Collaborative Filtering. In *Proc. of the 7th International Conference on Intelligent Systems Design and Applications*, (pp. 421–426). Rio de Janeiro, Brazil.

11. de Castro, P. A. D., de França, F. O., Ferreira, H. M., & Von Zuben, F. J. (2007). Evaluating the Performance of a Biclustering Algorithm Applied to Collaborative Filtering: A Comparative Analysis. In *Proc. of the 7th International Conference on Hybrid Intelligent Systems*, (pp. 65–70). Kaiserslautern, Germany.

12. de Castro, P. A. D., de França, F. O., Ferreira, H. M., & Von Zuben, F. J. (2007). Applying Biclustering to Text Mining: An Immune-Inspired Approach. In L. N. de Castro, F. J. Von Zuben, & H. Knidel (Eds.) *Artificial Immune Systems, Proc. of the 6th International Conference on Artificial Immune Systems (ICARIS)*, vol. 4628 of *Lecture Notes in Computer Science*, (pp. 83–94). Santos, Brazil.

13. de França, F. O., Bezerra, G. B. P., & Von Zuben, F. J. (2006). New Perspectives for the Biclustering Problem. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, (pp. 753–760). IEEE Press.

Apêndice A

Construção da base de dados artificiais

Para construir a base de dados, alguns aspectos de projeto foram estabelecidos. Primeiramente, uma forma de obter correlação total entre dois objetos, em um determinado conjunto de atributos, é definindo os valores desses atributos de tal forma que esses dois objetos se diferenciem por um mesmo valor constante em todos os atributos do conjunto. Sendo assim, os valores v_1^a do primeiro objeto e v_2^a do segundo objeto se correlacionam tal como:

$$v_1^a = v_2^a + b, \forall a \in A, \tag{A.1}$$

onde b é um valor constante denominado *bias* e A é o conjunto de atributos. Isso garante que a correlação seja total, representando uma coerência em relação às respostas.

A base de dados é composta por 1.000 objetos e 100 atributos, sendo que esses objetos poderão pertencer a um ou mais grupos distintos dentre quatro aqui definidos. Os quatro grupos são referenciados na sequência dos atributos, ou seja, o grupo 1 é definido pelos atributos de 1 a 25, o grupo 2 pelos atributos de 26 a 50, o grupo 3 de 51 a 75 e o grupo 4 de 76 a 100. Definidos os grupos, os objetos que pertencem a eles foram escolhidos de forma aleatória (notando que essa aleatoriedade não tornará os dados incertos) de tal forma que 100 objetos participam exclusivamente do grupo 1, 150 objetos participam exclusivamente do grupo 2, 50 objetos são exclusivos do grupo 3 e 200 objetos são exclusivos do grupo 4.

Os outros 500 objetos pertencem a mais de um grupo simultaneamente, sendo 150 objetos pertencentes aos grupos 1 e 2, 50 objetos pertencentes aos grupos 2 e 3, 200 objetos aos grupos 3 e 4 e 100 objetos aos grupos 2, 3 e 4. Uma vez definida a quantidade de objetos por grupo, é necessário definir de quais grupos cada objeto irá participar. Para isso, de forma a simplificar o processo, os objetos foram atribuídos sequencialmente aos grupos, na mesma ordem descrita anteriormente, primeiro os objetos com exclusividade de um grupo e depois aqueles que pertencem a mais de um grupo.

Finalmente, é necessário atribuir os valores para cada objeto. Para tanto, é necessário controlar os

valores que relacionam os objetos a um mesmo grupo, de forma que a relação definida na Eq. A.1 seja verdadeira. Para esse fim, gera-se um centroide que será responsável por definir um objeto qualquer dentro de um grupo, sendo que os valores deste oscilem em torno de $4,5$, sem muita intensidade de variação. Isso pode ser feito utilizando um gerador de números aleatórios com distribuição normal e calculando:

$$x_i = 4,5 + 0,1N(0,1), \forall i \in I \tag{A.2}$$

onde I representa o conjunto de atributos em que o vetor de centro está sendo gerado e N(0,1) é uma distribuição normal de média zero e variância unitária.

Tendo o vetor de centro, para gerar um objeto que pertença ao grupo avaliado no momento, basta somar um *bias* aleatório a esse vetor. Esse *bias* pode ser tanto positivo quanto negativo mas de tal forma que os valores do vetor resultante continuem dentro da faixa de valores $[4,5]$. Escolhendo um *bias* para cada objeto, esses são somados ao centroide, gerando objetos parciais, uma vez que leva-se em conta apenas um sub-conjunto de atributos para determinado grupo.

Após todos os valores dos objetos pertencentes a cada grupo terem sido gerados, o restante da matriz é preenchida da mesma forma, só que com valores variando em torno de 1, dentro de cada bloco de grupos, de tal forma que os objetos formem grupos com atributos de valores baixos, mas com correlação baixa em relação aos grupos com atributos de valores altos. Essa base de dados artificiais será denominada de *ArtDataset*.

A.1 Base de dados ruidosa

Para tornar a base ArtDataset um pouco mais próxima de uma base real, seus valores serão acrescidos de um valor aleatório com distribuição normal $N(0,1)$, definido para cada elemento, de tal forma que objetos de um mesmo grupo não tenham uma correlação perfeita mas, ainda assim, tenham uma correlação alta em relação a objetos de grupos distintos. Essa nova base será denominada *ArtDatasetRuido* e será utilizada para gerar as bases de dados com valores faltantes.

Essa base também será utilizada para estudar quais propriedades estatísticas os métodos de imputação tendem a preservar, de ArtDataset ou de ArtDatasetRuido. Isso porque algumas metodologias tendem a procurar preencher os valores de forma a estimar as quantidades estatísticas de uma base sem ruído e com grupos bem definidos.

A.2 Produção de dados faltantes

O próximo passo para a construção da biblioteca de bases para testes é a definição de bases com os três tipos existentes de dados faltantes. Para tanto, a probabilidade será definida de acordo com os modelos descritos na Seção 2.2.

A.2.1 MCAR

Para gerar uma base com dados faltantes do tipo MCAR, foi definida a retirada de 10% dos valores de cada objeto, escolhidos aleatoriamente por uma distribuição uniforme. Os valores ausentes foram marcados com o valor 0 e a base de dados denominada *ArtDatasetMCAR*. Apesar dessa escolha de percentual de dados faltantes, testes com um percentual maior serão efetuados, quando pertinente.

A.2.2 MAR

Para definir os dados faltantes do tipo MAR, três motivos distintos para a ausência de valores foram escolhidos, de forma a testar a capacidade de um método de lidar com motivos distintos na ausência dos dados e separar os modelos de imputação quando necessário. O *motivo 1* da ausência é quando objetos pertencentes aos grupos 1 e 2 tendem a omitir respostas correspondentes ao atributo 80 do grupo 4 de atributos, com probabilidade de 12%. Similarmente, objetos que pertencem aos grupos 2, 3 e 4 tendem a omitir respostas referentes ao atributo 95 do grupo 4, e esse será o *motivo 2*, que ocorre com probabilidade de 10%. Finalmente, o *motivo 3* é definido quando objetos pertencentes aos grupos 2 e 3 tendem a omitir respostas referentes ao atributo 65 do grupo 3, com probabilidade de 25%. A base gerada será denominada *ArtDatasetMAR*.

A.2.3 NMAR

Como último estudo de caso, os dados faltantes do tipo NMAR serão definidos com a probabilidade de uma certa resposta ser omitida, dado o valor dela própria. Dois motivos serão gerados para esse caso, sendo o primeiro motivo a ausência da variável 85 do grupo 4 de atributos toda vez que seu valor for maior que 4, com probabilidade de 60%. O segundo caso será a ausência da variável 30 do grupo 2 toda vez que essa tiver um valor abaixo de $2,5$, com probabilidade de 35%. A base de dados será denominada *ArtDatasetNMAR*