UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Tratamento de Dados Faltantes Empregando Biclusterização com Imputação Múltipla

Rosana Veroneze

Orientador: Prof. Dr. Fernando José Von Zuben
Co-orientador: Dr. Fabrício Olivetti de França

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de Concentração: Engenharia de Computação

Campinas – SP – Brasil

Junho de 2011

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

Veroneze, Rosana
V599t Tratamento de dados faltantes empregando biclusterização com imputação múltipla / Rosana Veroneze. --Campinas, SP: [s.n.], 2011.

Orientadores: Fernando José Von Zuben, Fabrício Olivetti de França.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

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

Título em Inglês: Treatment of missing data using biclustering with multiple imputation
Palavras-chave em Inglês: Missing data (Statistics), Recommender systems, Cluster, Evolutionary algorithms, Data mining
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: André Luís Vasconcelos Coelho, Romis Ribeiro de Faissol Attux
Data da defesa: 22-06-2011
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Rosana Veroneze

Data da Defesa: 22 de junho de 2011

Título da Tese: "Tratamento de Dados Faltantes Empregando Biclusterização com Imputação Múltipla"

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

Prof. Dr. André Luís Vasconcelos Coelho: acelc

Prof. Dr. Romis Ribeiro de Faissol Attux: romis

Resumo

As respostas fornecidas por sistemas de recomendação podem ser interpretadas como dados faltantes a serem imputados a partir do conhecimento dos dados presentes e de sua relação com os dados faltantes. Existem variadas técnicas de imputação de dados faltantes, sendo que o emprego de imputação múltipla será considerado neste trabalho. Também existem propostas alternativas para se chegar à imputação múltipla, sendo que se propõe aqui a biclusterização como uma estratégia eficaz, flexível e com desempenho promissor. Para tanto, primeiramente é realizada a análise de sensibilidade paramétrica do algoritmo SwarmBcluster, recentemente proposto para a tarefa de biclusterização e já adaptado, na literatura, para a realização de imputação única. Essa análise mostrou que a escolha correta dos parâmetros pode melhorar o desempenho do algoritmo. Em seguida, o SwarmBcluster é estendido para a implementação de imputação múltipla, sendo comparado com o bem-conhecido algoritmo NORM. A qualidade dos resultados obtidos é mensurada através de métricas diversas, as quais mostram que a biclusterização conduz a imputações múltiplas de melhor qualidade na maioria dos experimentos.

Palavras-chave: Dados faltantes, sistemas de recomendação, imputação de dados, imputação múltipla, biclusterização, otimização por colônia de formigas.

Abstract

The answers provided by recommender systems can be interpreted as missing data to be imputed considering the knowledge associated with the available data and the relation between the available and the missing data. There is a wide range of techniques for data imputation, and this work is concerned with multiple imputation. Alternative approaches for multiple imputation have already been proposed, and this work takes biclustering as an effective, flexible and promising strategy. To this end, firstly it is performed a parameter sensitivity analysis of the SwarmBcluster algorithm, recently proposed to implement biclustering and already adapted, in the literature, to accomplish single imputation of missing data. This analysis has indicated that a proper choice of parameters may significantly improve the performance of the algorithm. Secondly, SwarmBcluster was extended to implement multiple imputation, being compared with the well-known NORM algorithm. The quality of the obtained results is computed considering diverse metrics, which reveal that biclustering guides to imputations of better quality in the majority of the experiments.

Keywords: Missing data, recommender systems, data imputation, multiple imputation, biclustering, ant colony optimization.

Agradecimentos

Ao Prof. Dr. Fernando José Von Zuben, pelos ensinamentos, desafios e orientação.

Ao Dr. Fabrício Olivetti de França, pelo grande auxílio em minhas pesquisas e co-orientação.

Aos membros da banca, pelas contribuições e comentários junto ao texto da dissertação.

Ao Prof. Ms. Eduardo Nicola Zagari, pelos estímulos e desafios durante a graduação e por me apresentar ao Programa de Pós-graduação da Unicamp.

À Faculdade de Engenharia Elétrica e de Computação e ao Departamento de Engenharia de Computação e Automação Industrial, pelo fornecimento de instalações e condições de trabalho apropriadas.

À FATEC Indaiatuba, pelo apoio ao meu projeto, em especial à minha coordenadora Profa. Ms. Graça.

Aos meus colegas do LBiC, pelas críticas, sugestões e companheirismo.

À minha mãe, Irene, pelo amor incondicional e doação à nossa família.

Ao meu pai, Rosivaldo, pelo exemplo de franqueza, coragem e determinação.

Aos meus irmãos, Renata e Junior, meus parceiros desde a infância, por todos os aprendizados que tivemos e temos juntos.

Aos meus avós, por me fazerem sentir tão amada e feliz.

Ao meu priminho, Gustavo, por todas as palavras, brincadeiras, choros e sorrisos.

Ao meu noivo, Saullo, por todo amor, companheirismo, apoio e compreensão.

Aos meus queridos, Andreza, Cristiane, Fábio, Gisele Guio, Giselle Castro, Jeremias, Juliana, Márcia, Michele, Murillo, Vanessa e Viviane, pela grande amizade.

Ao meu amigo e professor de Ciências da 5ª série, Rogério Serra, por ter me cativado e cobrado o melhor de mim.

Agradeço, enfim, a Deus, pela paz em meu coração.

Aos meus avós

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ACO - Otimização por Colônias de Formigas

BEC - Biclusters com Evoluções Coerentes

BVC - Biclusters com Valores Constantes

BVCC - Biclusters com Valores Constantes nas Colunas

BVCL - Biclusters com Valores Constantes nas Linhas

BVCo - Biclusters com Valores Coerentes

CC - Algoritmo de Cheng & Church

CD - Cold-Deck

CMI - Imputação de Médias Condicionadas

HD - Hot-Deck

IU - Imputação Única

KDD - Descoberta de Conhecimento a partir de Bases de Dados

K-NN - K-Vizinhos mais Próximos

LD - Caso Completo

LVCF - Último Valor Carregado para Frente

MAE - Erro Absoluto Médio

MAR - Missing at Random

MCAR - Missing Completely at Random

MI - Imputação Múltipla

ML - Máxima Verossimilhança

MNAR - Missing Not at Random

MSR - Resíduo Quadrático Médio

NVCB - Próximo Valor Carregado para Trás

PD - Casos Disponíveis

RMSE - Erro Quadrático Médio

rSVD - Regulated Singular Value Decomposition

SEM - Modelagem de Equações Estruturais

SwarmBcluster - Swarm Intelligence Biclustering

TSP - Problema do Caixeiro Viajante

Capítulo 1

Introdução

Atualmente, muitos dados estão sendo coletados devido à facilidade na automação de processos, do uso generalizado de computadores e da alta capacidade de armazenamento das mídias (BROWN & KROS, 2003). Com o crescimento explosivo de dados disponíveis oriundos de várias fontes, a questão sobre a garantia da qualidade dos dados torna-se cada vez mais importante (WU *et al.*, 2004). Entretanto, a maioria dos bancos de dados existentes é caracterizada pela imprecisão e pela incompletude, isto é, pela presença de valores ruidosos e faltantes, respectivamente (FARHANGFAR *et al.*, 2004). Existem muitos motivos que levam a este cenário, dentre eles podem ser citados: entrada de dados manual (LAKSHMINARAYAN *et al.*, 1999; FARHANGFAR *et al.*, 2004; FARHANGFAR *et al.*, 2007), medições incorretas (FARHANGFAR *et al.*, 2004; FARHANGFAR *et al.*, 2007), equipamentos com falhas operacionais (WU *et al.*, 2004; FARHANGFAR *et al.*, 2004; FARHANGFAR *et al.*, 2007; BUUREN *et al.*, 1994; COLANTONIO *et al.*, 2010), alto custo de coleta de dados (MYRTVEIT *et al.*, 2001) e participantes de pesquisa que se negam a responder algum item de um questionário (MCKNIGHT *et al.*, 2007; ALLISON, 2001; BUUREN *et al.*, 1994; COLANTONIO *et al.*, 2010). Em muitas áreas, não é raro encontrar bases de dados que têm 50%, ou até mais, de suas entradas ruidosas ou ausentes (FARHANGFAR *et al.*, 2007).

Tratando especificamente do caso de falta de dados, três tipos de problemas estão associados aos valores faltantes: 1) perda de eficiência; 2) complicações na manipulação e análise dos dados; e 3) viés, resultante das discrepâncias entre os valores atribuídos aos dados faltantes e os valores reais desconhecidos (FARHANGFAR *et al.*, 2007). O não tratamento ou o tratamento inadequado dos dados faltantes também pode afetar a generalização dos resultados da análise (MCKNIGHT *et al*., 2007; COLANTONIO *et al*., 2010).

Apesar disso, e mesmo sabendo que os dados faltantes existem desde os primórdios das atividades de coleta de dados (GRAHAM, 2009; BROWN & KROS, 2003), pouca atenção vem sendo dada a eles. Muitos pesquisadores utilizavam – e ainda utilizam – procedimentos analíticos que

não foram concebidos para tratar adequadamente bases de dados com valores faltantes (SCHAFER & GRAHAM, 2002). Estes procedimentos foram concebidos para trabalhar com bases de dados completas (bases de dados que não possuem nenhum dado faltante). Outros pesquisadores utilizam métodos muito genéricos para preencher os dados faltantes (FARHANGFAR *et al.*, 2007), por exemplo, o preenchimento de um valor faltante numérico por zero.

Somente a partir de 1987, com a publicação dos livros: *Statistical Analysis with Missing Data*, de Little & Rubin, e *Multiple Imputation for Nonresponse in Surveys*, de Rubin, é que se passou a investir mais na análise dos dados faltantes (GRAHAM, 2009). Este fato, aliado ao advento da computação pessoal, formaram o alicerce para que softwares preparados para lidar com dados faltantes fossem desenvolvidos (GRAHAM, 2009). Também, neste mesmo ano, foram publicados três artigos de grande importância: *Estimation of linear models with incomplete data*, de ALLISON (1987); *On structural equation modeling with data that are not missing completely at random*, de MUTHÉN *et al.* (1987); e *The calculation of posterior distributions by data augmentation*, de TANNER & WONG (1987). Os dois primeiros artigos tratam do primeiro método acessível para lidar com dados faltantes, o qual utiliza *Modelagem de Equações Estruturais* (SEM – do inglês *Structural Equation Modeling*), e o terceiro artigo foi de grande importância para o desenvolvimento de softwares de *Imputação Múltipla* (GRAHAM, 2009).

Os dados faltantes tornaram-se um dilema ainda mais desafiador no processo de *Descoberta de Conhecimento a partir de Bases de Dados* (KDD – do inglês *Knowledge Discovery from Databases*), na medida em que quanto mais dados são coletados maior a probabilidade de existirem dados faltantes (BROWN & KROS, 2003). Evidentemente, os resultados extraídos de um processo de KDD são realmente úteis e aplicáveis apenas quando os dados de entrada são de alta qualidade (WU *et al.*, 2004). Os algoritmos clássicos de mineração de dados, por exemplo, podem encontrar dificuldades na identificação de agrupamentos reais se a base de dados possui dados faltantes. O mais provável é que esses algoritmos identifiquem apenas pequenos fragmentos desses agrupamentos (COLANTONIO *et al.*, 2010). Logo, para superar os desafios impostos pelos dados faltantes, atualmente estão sendo desenvolvidas diferentes estratégias para trabalhar com bases de dados que contêm valores faltantes (MYRTVEIT *et al.*, 2001).

É certo que evitar que os dados faltantes ocorram seria a melhor solução, e algumas estratégias podem ser adotadas, como, por exemplo, aumentar os benefícios dos participantes de

uma pesquisa (MCKNIGHT *et al.*, 2007) e repetir o experimento (COLANTONIO *et al.*, 2010). Entretanto, a prevenção de dados faltantes não é sempre possível ou viável e, neste caso, não existe outra maneira de se chegar a resultados confiáveis a não ser tratando-os. Os dados faltantes, geralmente, não são o foco principal de um estudo, mas são um incômodo comum, e tratá-los levanta dificuldades conceituais e desafios computacionais (SCHAFER & GRAHAM, 2002).

Existem vários métodos para o tratamento dos dados faltantes: entre eles, estão os métodos de imputação única, os métodos de máxima verossimilhança e os métodos de imputação múltipla (MI – do inglês *Multiple Imputation*). Os métodos de imputação única consistem, basicamente, em substituir os dados faltantes por valores factíveis. Os métodos de máxima verossimilhança objetivam a estimação dos parâmetros de modelos vinculados à distribuição dos dados. Enquanto os métodos de imputação única substituem cada valor faltante por um único valor, os métodos de MI substituem por x valores, com $x \geq 2$. Desse modo, são formadas x bases de dados completas que podem ser analisadas através de procedimentos convencionais. Os resultados dessas análises são agregados, gerando estimativas únicas para os parâmetros de interesse. Um parâmetro de interesse é uma informação que se deseja conhecer sobre uma população, como por exemplo, a média populacional.

Ao contrário dos métodos de imputação única, os métodos de MI não tendem a subestimar a variabilidade da amostra (LITTLE & RUBIN, 2002). Além disso, os métodos de MI são provavelmente menos sensíveis que os métodos de verossimilhança na escolha do modelo de distribuição dos dados, porque o modelo é usado somente para a imputação dos valores faltantes e não para estimar os parâmetros (ALLISON, 2001). Por isso, com o avanço da computação e com a proliferação de softwares que implementam algum método de MI, ela se tornou rapidamente a classe de métodos mais indicada para manipular dados faltantes (MCKNIGHT *et al.*, 2007).

Os softwares que implementam MI utilizam procedimentos estatísticos para realizar as x imputações. Normalmente, são algoritmos que usam técnicas de Monte Carlo em cadeias de Markov (SCHAFER, 1997), como o algoritmo chamado *Data Augmentation* (TANNER & WONG, 1987). Esses algoritmos requerem que sejam feitas, no mínimo, duas suposições. A primeira é sobre a distribuição dos dados e a segunda é sobre o mecanismo associado aos dados faltantes. Resumidamente, o mecanismo é o motivo da falta de dados, por exemplo, um evento aleatório. Como é difícil conhecer o mecanismo dos dados faltantes e a distribuição dos dados,

principalmente em bases de dados incompletas, é interessante considerar técnicas de imputação que não dependam dessas informações, mas que, ao mesmo tempo, respeitem as relações existentes entre os dados observados da base de dados. Uma técnica capaz de extrair a relação entre subconjuntos de atributos e objetos de uma base de dados, explorando a correlação única entre eles, é a biclusterização (CHENG & CHURCH, 2000).

Utilizar a biclusterização como modelo para a MI tem algumas vantagens. A primeira delas é não ser necessário considerar um modelo global para toda a base de dados. Uma vez que o processo de imputação ocorre apenas em um subconjunto menor e cuidadosamente selecionado de dados, a imputação sofre menos influência do ruído e da falta de outros dados, pois estes não participam de todo o processo de estimação. Além do mais, como os biclusters mostram explicitamente quais objetos e atributos foram selecionados, eles permitem interpretar e explicar diretamente os modelos e o mecanismo por trás da falta de dados (DE FRANÇA, 2010). Além disso, é possível explorar a capacidade da técnica de MI em estimar o impacto causado pelos dados faltantes nas estimativas dos parâmetros de interesse, para avaliar a qualidade das estimações realizadas.

Para trabalhar a MI juntamente com a biclusterização, foi utilizado um algoritmo de biclusterização, denominado SwarmBcluster, o qual é baseado em Inteligência de Enxames e foi recentemente proposto por DE FRANÇA & VON ZUBEN (2010). O primeiro passo para proceder com a técnica de MI no tratamento de um conjunto de dados incompleto é realizar $x \geq 2$ imputações para cada dado faltante, produzindo x conjuntos de dados completos. É justamente nesse passo que o SwarmBcluster atua. Realizando x execuções do algoritmo SwarmBcluster são produzidos os x conjuntos de dados completos necessários à técnica de MI.

Até o presente, o SwarmBcluster é o único algoritmo de biclusterização adaptado para trabalhar com a imputação de dados numéricos reais (DE FRANÇA, 2010). Na literatura, existe apenas outro algoritmo de biclusterização para imputação de dados faltantes, denominado ABBA (COLANTONIO *et al.*, 2010), mas que trabalha apenas com bases de dados binárias. Adicionalmente, existe outra aplicação, similar à imputação, chamada filtragem colaborativa (KOREN & BELL, 2011), que já foi resolvida por meio da biclusterização. Como exemplo, podem-se citar os trabalhos de DE CASTRO *et al.* (2007b), DE CASTRO *et al.* (2007d) e SYMEONIDIS *et al.* (2007). Mas, diferentemente da imputação, a filtragem colaborativa não requer que todos os dados faltantes sejam estimados. Ademais, o SwarmBcluster foi o primeiro algoritmo a

incorporar um modelo de coerência durante a estimação dos valores faltantes. O modelo de coerência descreve de que forma os valores dentro de um bicluster devem estar correlacionados entre si.

1.1 Objetivos

Como o SwarmBcluster é uma recente e promissora proposta, é necessário investigar seu comportamento com diferentes conjuntos de dados, percentuais e mecanismos de dados faltantes. Por isso, os objetivos da primeira parte dos experimentos realizados neste trabalho são: (*i*) investigar esse algoritmo, com o objetivo de indicar como escolher os valores dos seus parâmetros corretamente e como esses parâmetros influenciam os resultados finais; e (*ii*) mostrar como o SwarmBcluster é um algoritmo competitivo para tratar dados faltantes associados a qualquer mecanismo responsável por sua ausência. Em DE FRANÇA (2010), já foram realizados alguns experimentos que comparam o desempenho do SwarmBcluster com os algoritmos KNNImpute e rSVD frente ao mecanismo chamado *MCAR*. Além de este trabalho explorar todos os três mecanismos existentes, ele também utiliza uma nova versão do SwarmBcluster. Ao contrário da versão utilizada por DE FRANÇA (2010), esta nova versão leva em consideração os dados faltantes não cobertos por biclusters no cálculo do valor de *MAE* e *RMSE*, que são as duas métricas escolhidas para se medir o desempenho da técnica de imputação.

A segunda parte dos experimentos emprega a biclusterização para gerar as imputações múltiplas necessárias ao método de MI e os objetivos são: (*i*) investigar o comportamento da biclusterização sendo utilizada como modelo para a MI; (*ii*) avaliar o quanto se pode ganhar na qualidade da imputação quando um dado faltante é estimado com base em imputações múltiplas em vez de uma imputação única; e (*iii*) indicar como as métricas descritas por RUBIN (1987) para avaliar a qualidade da MI podem ajudar a medir a qualidade da estimação dos valores faltantes.

1.2 Organização do texto

Os primeiros capítulos desta dissertação realizam a revisão bibliográfica. No Capítulo 2, os principais conceitos relacionados aos dados faltantes são explicados. O Capítulo 3 apresenta alguns métodos que já se destacaram no tratamento de dados faltantes. No Capítulo 4, os principais conceitos relacionados à biclusterização são explicados. O Capítulo 5 descreve o

algoritmo SwarmBcluster, e também descreve o KNNImpute e o rSVD, os quais foram utilizados nos experimentos de comparação de desempenho para imputação única.

Os dois capítulos seguintes apresentam os experimentos realizados. O Capítulo 6 apresenta os experimentos de análise de sensibilidade paramétrica e os experimentos de comparação de desempenho para imputação única. O Capítulo 7 apresenta os experimentos realizados com biclusterização e MI.

Para finalizar, no Capítulo 8 são feitas as considerações finais e são apresentadas algumas perspectivas para trabalhos futuros nessa linha de pesquisa.

Capítulo 2

Dados faltantes

Este capítulo visa explanar os conceitos mais importantes relacionados aos dados faltantes. É importante conhecer tais conceitos para que os dados faltantes sejam manipulados adequadamente. Além disso, este capítulo também traz os principais conceitos sobre S*istemas de Recomendação*, que podem ser interpretados como um caso particular de dados faltantes.

2.1 Definição

Em um conjunto de dados, um objeto (também chamado de registro ou caso) é completo se todos os seus atributos (também chamados de campos ou variáveis) estão preenchidos com dados apropriados. Um *dado faltante* indica que um atributo de um objeto está vazio. Um *caso incompleto* é um objeto que contém dados faltantes (WU *et al.*, 2004). Segundo MCKNIGHT *et al*. (2007), "de um modo geral, o termo *dados faltantes* significa que está faltando algum tipo de informação sobre o fenômeno em que estamos interessados". Normalmente, são observações que deveriam ter sido feitas, mas não foram por algum motivo. Logicamente, quando isso acontece, a capacidade de entender a natureza do fenômeno pode ser reduzida e o impacto nos resultados dos estudos nem sempre são conhecidos, tornando-se difícil extrair um conhecimento útil a partir dos dados analisados (MCKNIGHT *et al*., 2007).

Um procedimento comum, também, é substituir valores ruidosos por valores faltantes, criando, assim, mais dados faltantes na base de dados (MYRTVEIT *et al.*, 2001). Os valores ruidosos são aqueles que apresentam um desvio significativo do valor real (MYRTVEIT *et al.*, 2001). Vários tipos de dados ruidosos são encontrados em um banco de dados, como: incorreção, duplicação e inconsistência (WU *et al.*, 2004).

Para que seja realizado um tratamento adequado dos dados faltantes, é importante que o analista de dados identifique algumas características desses dados, como: mecanismo, padrão e quantidade. Essas características serão estudadas nas seções subsequentes.

2.2 Mecanismos associados aos dados faltantes

Uma questão importante é saber se a presença de dados faltantes em uma variável está vinculada a algum processo identificável (LITTLE & RUBIN, 2002), ou seja, conhecer o mecanismo que gerou os dados faltantes. Este conhecimento irá auxiliar na escolha da técnica de tratamento mais adequada, sendo que a maioria delas requer que os dados faltantes sejam do tipo *ignorável* (MCKNIGHT *et al.*, 2007). Basicamente, o termo ignorável significa que não há necessidade de modelar o mecanismo como parte do processo de estimação (ALLISON, 2001).

O principal sistema de classificação foi criado por Donald Rubin em 1976. Para explicá-lo, considere A uma matriz de dados coletados, com M linhas, as quais representam os objetos, N colunas, as quais representam os atributos, e com $a_i = (a_{i1}, ..., a_{iN})$, onde a_{ij} é o valor do atributo j para o objeto i. Pode-se dividir A em dois conjuntos:

$$A = \{A_{obs}, A_{falt}\},$$

onde A_{obs} são os dados observados (não-faltantes) e A_{falt} são os dados faltantes.

Correspondente a cada matriz de dados A, existe um identificador de dados faltantes, uma matriz R de mesma dimensão de A, onde $r_{ij} = 1$, se a_{ij} é observado, e $r_{ij} = 0$, caso contrário.

O mecanismo de dados faltantes é caracterizado pela distribuição condicional de R dado A, $P(R \mid A)$, a qual pode ser de três tipos (LITTLE & RUBIN, 2002):

a) Missing Completely at Random

Se os dados faltantes não dependem dos valores de A, faltantes ou observados, tem-se:

$$P(R \mid A) = P(R). \qquad (2.1)$$

Isso significa que a causa que levou aos dados faltantes é um evento aleatório, ou seja, o mecanismo é *Missing Completely at Random* (MCAR). Nesse caso, os valores faltantes para uma variável são uma simples amostra aleatória dos dados dessa variável, ou seja, a distribuição dos valores faltantes é de mesma natureza da dos valores observados (ZHANG, 2003).

Em muitas situações, o mecanismo é MCAR mesmo se os dados faltantes existam devido a algum evento que não é verdadeiramente aleatório, mas sim causado por alguma variável. Isso acontece quando a causa é uma variável não correlacionada com a variável que possui valores faltantes (GRAHAM *et al.*, 1995).

A grande vantagem de o mecanismo ser MCAR é que a causa que levou aos dados faltantes não precisa fazer parte da análise para controlar a influência destes nos resultados da pesquisa (GRAHAM *et al.*, 1995). Por isso, ALLISON (2001) afirma que "embora MCAR seja uma suposição forte, há momentos em que é razoável, especialmente quando os dados estão em falta devido a decisões do processo de coleta". Por exemplo, isso pode ocorrer se uma variável de um estudo é muito custosa para se medir. Então, decide-se que ela será medida apenas para um subconjunto aleatório da amostra, o que implica nos dados serem MCAR para as demais variáveis dessa amostra (ALLISON, 2001).

É importante enfatizar que, apesar das vantagens associadas ao fato dos dados serem MCAR, não é qualquer método dentre os capazes de lidar com dados faltantes que produzirá bons resultados.

b) Missing at Random

Se os dados faltantes não dependem dos valores de A_{falt} e apenas dos valores de A_{obs}, tem-se:

$$P(R \mid A) = P(R \mid A_{obs}). \tag{2.2}$$

Este mecanismo é chamado *Missing at Random* (MAR). Neste, os dados faltantes são causados por alguma variável observada, disponível para análise e correlacionada com a variável que possui dados faltantes (GRAHAM *et al.*, 1995). Neste caso, os valores faltantes de uma variável são como uma amostra aleatória simples dos dados para essa variável dentro de subgrupos definidos por valores observados, e a distribuição dos valores faltantes é a mesma que a distribuição dos valores observados dentro de cada subgrupo (ZHANG, 2003).

Este é um mecanismo acessível, pois se a causa que levou aos dados faltantes pode ser medida e é incluída devidamente na análise, todas as influências causadas por eles podem ser consideradas (GRAHAM *et al.*, 1995).

c) Missing Not at Random

Quando a distribuição de R depende dos dados faltantes contidos na matriz A (A_{falt}), podendo também depender dos dados obsevados (A_{obs}), tem-se (ZHANG, 2003):

$$P(R \mid A) \neq P(R \mid A_{obs}). \tag{2.3}$$

Este mecanismo é chamado *Missing not at Random* (MNAR), ou também mecanismo inacessível. Esta situação pode acontecer quando a causa dos dados faltantes numa variável é o próprio valor dela.

A Figura 2.1 traz um resumo dos três mecanismos, evidenciando suas principais diferenças. O mecanismo é MCAR se os dados faltantes são causados por processos aleatórios. O mecanismo é MAR se os dados faltantes são causados por uma ou mais variáveis observadas. O mecanismo é MNAR se os dados faltantes são causados por uma ou mais variáveis não-observadas; além disso, pode ou não existir uma relação entre os dados observados e os dados faltantes quando o mecanismo é MNAR.

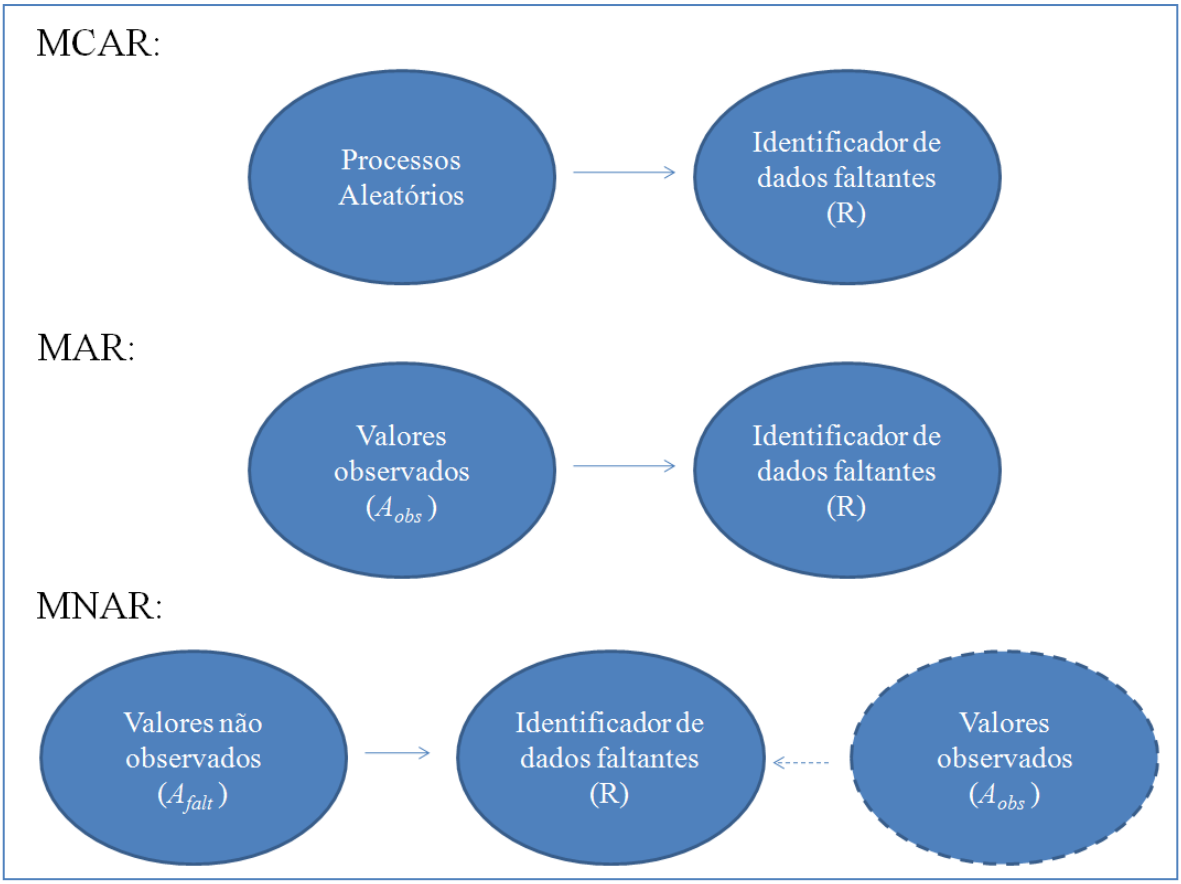


Figura 2.1: Mecanismos associados aos dados faltantes.
Fonte: MCKNIGHT *et al.*, 2007.

Os mecanismos MCAR e MAR também são chamados de *ignoráveis*, enquanto o mecanismo MNAR é também chamado de *não-ignorável* (GRAHAM, 2009). É importante

salientar que o termo ignorável não é utilizado para indicar que os pesquisadores podem ser indiferentes aos dados faltantes (MCKNIGHT *et al.*, 2007).

Os mecanismos ignoráveis são considerados mais fáceis de lidar, pois seus efeitos nos modelos estatísticos estão disponíveis para o analista de dados (MCKNIGHT *et al.*, 2007). O mecanismo MCAR não deve ter grande impacto na estimação dos parâmetros, pois os dados faltantes acontecem de maneira completamente aleatória. Quando o mecanismo é MAR, existe um processo sistemático subjacente à falta de dados que pode ser modelado através dos dados observados da matriz (MCKNIGHT *et al.*, 2007).

Ao contrário, quando o mecanismo é não-ignorável, não existe nenhuma informação dentro do conjunto de dados que permita modelar e compreender a maneira com que os dados faltantes aconteceram. Portanto, o efeito deste mecanismo é desconhecido e potencialmente perigoso, devendo ser modelado para que sejam obtidas boas estimativas dos parâmetros de interesse (MCKNIGHT *et al.*, 2007). Dessa forma, diagnosticar o mecanismo ajuda o pesquisador e o analista de dados a entender a natureza dos dados faltantes e o potencial impacto nos resultados dos estudos e nas interpretações destes (MCKNIGHT *et al.*, 2007).

Para realizar este diagnóstico, primeiramente verifica-se se o mecanismo é MCAR. Se não for, verifica-se se é MAR ou MNAR. Para verificar se é MCAR, existe um único método formal, proposto por LITTLE (1988). Através do teste chi-quadrado, ele propôs comparar as médias de um atributo para cada padrão de dados faltantes com as médias calculadas para todo conjunto de dados através de um método robusto de estimação de parâmetros. Antes desse método, os analistas de dados costumavam criar códigos fictícios para todas as variáveis com dados faltantes e, baseados nisso, definiam dois grupos para cada variável de interesse: o que contém dados faltantes e o que não contém. Os analistas, então, conduziam testes t comparando as médias de algumas ou de todas as variáveis remanescentes para cada um dos dois grupos. Uma diferença significativa entre estes dois grupos seria uma evidência contra o mecanismo MCAR.

Se o mecanismo não for MCAR, é necessário saber se o mecanismo que criou os dados faltantes é relacionado com as informações conhecidas e, infelizmente, não existe nenhum método formal para isso. Porém, segundo SCHAFER (1997), existem quatro situações em que se pode supor que o mecanismo é ignorável:

- Numa situação em que algumas informações são colhidas de todos os objetos da base de dados, e outras informações adicionais são colhidas apenas de um subgrupo da amostra

original, sendo que esse subgrupo é selecionado devido a alguma informação coletada para toda a amostra.

- Quando os pesquisadores podem substituir os objetos incompletos por outros completos, com as mesmas características.
- Em testes controlados aleatoriamente, em que o número de objetos, nas diferentes intervenções, é desigual ou desbalanceado devido a causas inesperadas e não devido a um processo sistemático.
- Se informações são colhidas de uma amostra e, posteriormente, informações adicionais são colhidas de um subgrupo selecionado aleatoriamente, ou selecionado baseado nas informações colhidas previamente.

Se uma dessas situações acima ocorrer, o mecanismo pode ser considerado MAR, senão deve ser considerado MNAR.

Tabela 2.1: Mesclando os sistemas de classificação para dados faltantes.

	MCAR	**MAR**	**MNAR**
Variável (Item)	Sujeitos omitem respostas aleatoriamente	Sujeitos omitem respostas que podem ser conseguidas por outras respostas	Sujeitos não respondem itens indiscriminadamente
Indivíduos ou sujeitos	Faltam dados de sujeitos aleatoriamente	Faltam dados de sujeitos, mas que são relacionados com os dados demográficos disponíveis	Faltam dados de sujeitos e são relacionados com os dados demográficos não medidos
Ocasiões	Sujeitos aleatoriamente não se apresentam na sessão	Sujeitos que se desempenham mal na sessão anterior, não se apresentam na sessão seguinte	Sujeitos que estão se desempenhando mal na sessão atual, deixam de participar

Fonte: MCKNIGHT *et al*., 2007.

Este sistema de classificação criado por Rubin foca na estrutura dos dados faltantes (relação entre R, A_{obs} e A_{falt}). MCKNIGHT *et al*. (2007) propuseram relacionar a classificação de Rubin com a classificação de CATTELL (1966), chamada *Dimensões* de dados faltantes, com o intuito de proporcionar um método mais informativo. Esta classificação é formulada em termos das dimensões de uma "caixa de dados". Cada uma das três dimensões da caixa representa uma

faceta diferente dos dados: (1) variáveis/itens, (2) indivíduos, e (3) ocasiões (MCKNIGHT *et al.*, 2007). A Tabela 2.1 traz esta relação. Supõe-se aqui que a coleta de dados envolve respostas fornecidas por participantes de uma pesquisa, numa entrevista.

2.3 Padrão dos dados faltantes

Para escolher o procedimento mais adequado para tratar os dados faltantes, também é importante detectar o padrão dos dados faltantes, o qual descreve que valores foram observados e que valores estão faltantes numa matriz de dados (LITTLE & RUBIN, 2002). Através do padrão, é possível identificar se há ou não consistência no modo pelo qual os dados não foram observados (MCKNIGHT *et al.*, 2007). Dessa forma, detectar o padrão auxilia na descoberta do mecanismo, uma vez que este se preocupa com a relação existente entre os dados faltantes e observados (LITTLE & RUBIN, 2002).

Novamente, pode-se utilizar a matriz *R* para, agora, observar o padrão dos dados faltantes. Por exemplo, a Figura 2.2 exibe uma matriz R_{8x5} e o padrão de dados faltantes que se pode detectar através dela. Esse padrão é chamado de *Univariado*. É um padrão consistente, único e detectável. Este padrão pode acontecer, por exemplo, se alguns participantes de uma pesquisa não respondem a um item do questionário.

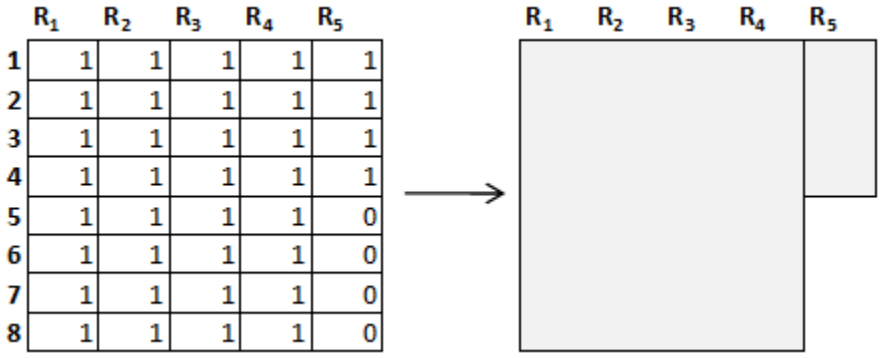


Figura 2.2: Padrão de dados faltantes *Univariado*.

A Figura 2.3 traz mais dois exemplos de padrões de dados faltantes. Uma matriz de dados tem o padrão *Monótono* (Figura 2.3.a) se, sempre que um elemento a_{ij} é faltante, os elementos a_{ik} também são faltantes, para todo $k > j$ (LITTLE & RUBIN, 2002). Este padrão é comum em estudos

longitudinais (método de pesquisa que visa analisar as variações nas características dos mesmos elementos amostrais ao longo de um período de tempo) e pode acontecer, por exemplo, se pacientes participantes de uma pesquisa desistem de participar. Já a Figura 2.3.b traz o padrão *Arbitrário*, que acontece quando alguns itens não são respondidos de forma aleatória.

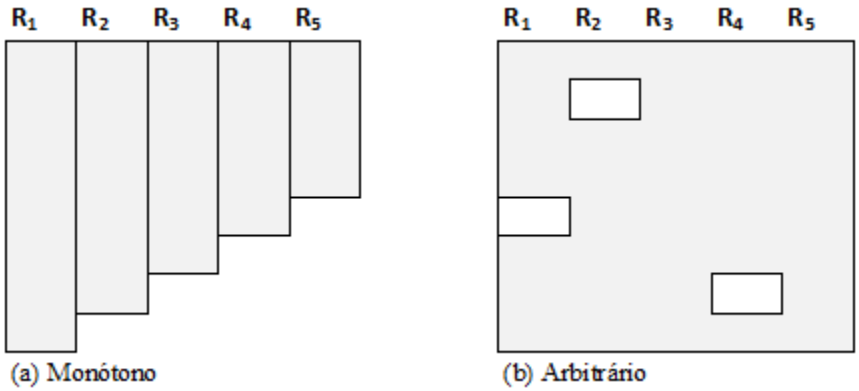


Figura 2.3: Mais exemplos de padrões de dados faltantes.

Como se observa, o padrão indica se os dados faltantes ocorrem de forma não-estruturada ou sistemática. Os dados faltantes são não-estruturados quando existem múltiplos padrões de dados faltantes entre os objetos do estudo, o que indica que o mecanismo pode ser aleatório (MCKNIGHT *et al.*, 2007). Em contrapartida, se os dados faltantes são sistemáticos, ou seja, estruturados ou capazes de expressar alguma tendência, há um indicador de que o mecanismo que governa os dados faltantes não é aleatório (MCKNIGHT *et al.*, 2007). Além do mais, o padrão ajuda a identificar quais métodos são factíveis para o tratamento dos dados faltantes (MCKNIGHT *et al.*, 2007).

2.4 Quantidade de dados faltantes

A quantidade de dados faltantes é um conceito ambíguo, pois, por exemplo, pode se referir à quantidade de:

- objetos que possuem dados faltantes;
- atributos que possuem dados faltantes;
- valores faltantes em uma atributo específico;
- valores faltantes em um conjunto específico de atributos; e

- valores faltantes de todo o conjunto de dados.

Esta também é uma informação importante para estimar a eficácia dos procedimentos estatísticos que se pretende usar. O método *Listwise Deletion* (elucidado na Seção 3.1), por exemplo, não é recomendado para grandes porcentagens de objetos incompletos, devido ao grau de redução no tamanho da amostra que sua aplicação traz.

A quantidade de dados faltantes também está fortemente relacionada com a precisão dos parâmetros a serem estimados: quanto maior a quantidade, mais difícil será chegar a bons resultados. Mas também é importante observar a qualidade dos dados faltantes, pois uma grande quantidade de dados faltantes, sob um mecanismo ignorável, pode ser bem mais facilmente tratada do que uma pequena quantidade sob o mecanismo MNAR (MCKNIGHT *et al*., 2007).

2.5 A seleção do método apropriado

Como já foi dito, fazer um esforço para prevenir os dados faltantes é mais produtivo do que tratá-los. Mas, mesmo prevenindo, será muito difícil eliminá-los completamente, sendo necessário tratá-los. Porém, antes de iniciar o tratamento, é importante descobrir algumas informações que serão de grande valia na escolha das técnicas mais apropriadas para tratar os dados faltantes. MCKNIGHT *et al.* (2007) dividem este processo em algumas tarefas, as quais serão descritas a seguir.

O primeiro passo é identificar quais atributos do conjunto de dados são realmente relevantes à análise que será feita e, desta maneira, simplificar o diagnóstico e o tratamento dos dados faltantes, devido à redução do número de atributos que serão analisados.

Se os dados possuem uma estrutura hierárquica (como micro e macro-unidades), o segundo passo é especificar em que nível a análise será feita. As macro-unidades são formadas por agregações ou grupos de micro-unidades, por exemplo: valores dos itens (micro) versus valores dos fatores (macro) em uma análise fatorial. Identificar o nível da análise também facilitará no tratamento dos dados faltantes, pois se a análise será feita no nível macro, dados faltantes no nível micro podem não ser tão relevantes (e vice-versa).

O terceiro passo é realizar o diagnóstico dos dados faltantes, isto é, identificar o mecanismo, o padrão e a quantidade de dados faltantes. Como já foi mencionado, esse

diagnóstico produzirá entendimento sobre os dados faltantes, auxiliando no processo de tomada de decisão.

Também é necessário ter alguma familiaridade com os requisitos e suposições que as diversas técnicas que lidam com dados faltantes carregam. Muitos desses requisitos e suposições estão intimamente relacionados com as informações e características dos dados faltantes, que foram levantadas nas etapas anteriores. O Capítulo 3 irá formalizar alguns dos principais métodos de tratamento de dados faltantes. Mas, antes, a próxima seção deste trabalho traz os principais conceitos sobre os sistemas de recomendação, que podem ser vistos como um caso particular e muito importante de dados faltantes.

2.6 Sistemas de Recomendação

O objetivo dos sistemas de recomendação é ajudar os *usuários* a encontrar *itens* que eles podem apreciar (CANDILLIER *et al.*, 2008). Esses itens podem ser, por exemplo, filmes, músicas, livros, páginas da web, notícias, textos com temas específicos, restaurantes, roteiros de férias, amigos em uma rede social ou serviços financeiros. Os sistemas de recomendação ajudam os usuários a encontrar tais itens de interesse baseados em alguma informação sobre seu histórico de preferências (CANDILLIER *et al.*, 2008). Esses itens passíveis de serem sugeridos são as informações faltantes que se deseja descobrir. Normalmente, os sistemas de recomendação empregam algoritmos de *imputação única,* abordagem de tratamento de dados faltantes a ser descrita na Seção 3.4, para encontrar tais itens de interesse.

Os sistemas de recomendação representam uma área de pesquisa que se tornou importante desde os primeiros artigos que foram publicados em meados dos anos de 1990 (ADOMAVICIUS & TUZHILIN, 2005). Baseando-se em como as sugestões podem ser feitas, essa área foi subdividida em três: filtragem colaborativa, filtragem baseada em conteúdo e filtragem híbrida. Esses três tipos de sistemas de recomendação são descritos a seguir.

2.6.1 Filtragem colaborativa

Em filtragem colaborativa, a entrada do sistema é, basicamente, um conjunto de avaliações atribuídas a itens de interesse pelos usuários. As sugestões podem ser feitas com base nos

usuários que compartilham preferências em comum (*baseada em usuário*) ou com base em itens similares (*baseada em item*).

Na abordagem baseada em usuário, os usuários são comparados com base na apreciação que compartilham dos itens, criando a noção de *vizinhança de usuários*. A predição da avaliação de um item por um usuário é baseada nas avaliações feitas por outros usuários que fazem parte de sua vizinhança. Logo, é necessário estabelecer: (*i*) uma medida de similaridade entre os usuários com base nas avaliações que já fizeram; (*ii*) o tamanho da vizinhança; e (*iii*) um método para combinar as avaliações realizadas pelos vizinhos sobre o item de interesse.

A abordagem baseada em item é similar à abordagem baseada em usuário. Dada uma medida de similaridade entre os itens, tais métodos primeiramente definem a *vizinhança dos itens*. A predição da avaliação de um item por um usuário é obtida utilizando as avaliações desse usuário na vizinhança desse item.

Mais recentemente, uma terceira abordagem foi proposta por BREESE *et al*. (1998), chamada de *baseada em modelo*. A ideia geral é derivar um modelo dos dados (off-line) para predizer as avaliações on-line, o mais rápido possível (CANDILLIER *et al.*, 2008). Um exemplo é o uso de técnicas de agrupamento (BERKHIN, 2006) para agrupar os usuários e, então, predizer a avaliação de um item por um usuário com base nas avaliações dos outros usuários que pertencem ao mesmo grupo (UNGAR & FOSTER, 1998; O'CONNER & HERLOCKER, 1999). Outros modelos também foram estudados, como o uso de regras de associação (SARWAR *et al*., 2000; LIN *et al*., 2002) e o uso de biclusterização (DE FRANÇA, 2010).

Dois problemas claros da filtragem colaborativa são a adição de um novo usuário ou a adição de um novo item ao conjunto de dados, pois é impossível definir a vizinhança ou o modelo adequado de um caso incompleto. Um usuário precisa avaliar uma quantidade suficiente de itens para que o sistema possa encontrar uma vizinhança, ou um modelo adequado, para ele. O mesmo vale para o item, ou seja, é preciso que ele tenha sido avaliado por uma quantidade suficiente de usuários para que o sistema possa encontrar uma vizinhança, ou um modelo adequado, para ele.

Os algoritmos KNNImpute (CANDILLIER *et al.*, 2008), rSVD (FUNK, 2006) e SwarmBcluster (DE FRANÇA, 2010), a serem descritos no Capítulo 5, são exemplos de algoritmos de filtragem colaborativa.

2.6.2 Filtragem baseada em conteúdo

Sistemas de recomendação baseados em conteúdo recomendam um item para um usuário com base na descrição do item e no perfil de interesse do usuário. Logicamente, faz-se necessário: (*i*) um meio para descrever os itens; (*ii*) um meio de descrever os perfis dos usuários; e (*iii*) um meio de comparar a descrição dos itens com o perfil do usuário para determinar o que sugerir.

A descrição do item depende do tipo do item que está sendo sugerido. Por exemplo, se os itens se referem a filmes, as características poderiam ser: gênero, diretor e atores principais.

Um *perfil de usuário* pode ser construído *explicitamente* ou *implicitamente*. Explicitamente através de questionários sobre suas preferências com relação às características dos itens. Por exemplo, uma das perguntas do questionário poderia ser o quanto o usuário aprecia o gênero drama. Se o usuário disser que aprecia muito, o sistema poderia sugerir a ele filmes de drama. O perfil do usuário também pode ser construído implicitamente, com base em suas ações passadas, ou seja, procurando aspectos em comum nos itens que ele gostou e aspectos em comum nos itens que ele não gostou.

Existem três problemas associados à filtragem baseada em conteúdo (ADOMAVICIUS & TUZHILIN, 2005). O primeiro deles é a análise de conteúdo limitada, pois essas técnicas são limitadas pelas características que são associadas aos itens. Uma das opções é inserir as características dos itens manualmente, mas isso nem sempre é viável. Logo, outra opção é extrair automaticamente as características dos itens. Em contrapartida, alguns domínios, como os dados multimídia, possuem um problema inerente com a extração automática de características. Ademais, se dois itens diferentes forem representados pelo mesmo conjunto de características, eles podem ser indistinguíveis. O segundo problema é a superespecialização, isto é, limitar as sugestões a itens similares àqueles que já foram avaliados, o que pode tornar as sugestões monótonas. O terceiro e último problema é a adição de um novo usuário, pois ele precisa avaliar uma quantidade suficiente de itens para que o sistema estabeleça o seu perfil.

O algoritmo *K*-Vizinhos mais Próximos (FRIEDMAN *et al.*, 1977) é frequentemente utilizado como um método de filtragem baseada em conteúdo, utilizando as características conhecidas dos itens para determinar a semelhança entre eles.

2.6.3 Filtragem Híbrida

Como o próprio nome diz, na filtragem híbrida, tanto a filtragem colaborativa como a filtragem baseada em conteúdo são utilizadas. Uma maneira simples de realizar a filtragem híbrida é realizar a filtragem colaborativa e a filtragem baseada em conteúdo de maneira independente e, em seguida, combinar os resultados obtidos. Entretanto, várias propostas mais elaboradas foram sugeridas na literatura. Em POLCICOVA *et al*. (2000) e MELVILLE *et al*. (2002), por exemplo, a matriz com as avaliações é enriquecida com predições baseadas em conteúdo e, em seguida, é utilizada a filtragem colaborativa.

A técnica de filtragem colaborativa é a mais implementada e a que, na maioria das vezes, gera melhores resultados (CANDILLIER *et al.*, 2008). A filtragem baseada em conteúdo é utilizada, normalmente, para transpor os limites impostos pela filtragem colaborativa e como uma maneira natural de interagir com os usuários.

2.7 Síntese do Capítulo

Este capítulo apresentou os principais conceitos relacionados aos dados faltantes. Inicialmente, o problema dos dados faltantes foi definido. Em seguida, foram apresentadas três características importantes relacionadas aos dados faltantes, que auxiliam na escolha de um método adequado para o tratamento dos dados faltantes.

A primeira dessas características é o mecanismo associado aos dados faltantes, que pode ser: MCAR, MAR ou MNAR. O mecanismo MCAR ocorre quando a causa que gerou os dados faltantes é um evento completamente aleatório. O mecanismo MAR ocorre quando os dados faltantes presentes em um atributo dependem dos valores observados em outro(s) atributo(s). E o mecanismo MNAR ocorre quando os dados faltantes presentes em um atributo dependem de seu próprio valor.

A segunda dessas características é o padrão. O padrão indica se os dados faltantes ocorrem de forma não-estruturada ou sistemática. Se os dados faltantes são não-estruturados, o mecanismo pode ser aleatório (MCAR). Caso contrário, há um forte indicativo de que o mecanismo que governa os dados faltantes não é aleatório.

A terceira e última dessas características é a quantidade de dados faltantes. A quantidade de dados faltantes é um conceito ambíguo, pois, por exemplo, pode-se referir à quantidade de dados em todo o conjunto de dados ou à quantidade de dados faltantes em um atributo específico.

Com base nessas características e com alguma familiaridade com os requisitos e suposições que as diversas técnicas que lidam com dados faltantes carregam, é possível selecionar a técnica apropriada. Alguns dos principais métodos para o tratamento dos dados faltantes são apresentados no próximo capítulo.

Para concluir este capítulo, um caso particular que envolve dados faltantes foi apresentado: os sistemas de recomendação. De forma bem resumida, os sistemas de recomendação precisam predizer as informações inexistentes com base nas informações existentes, pois são essas predições que vão indicar os itens a serem sugeridos. Os métodos de *imputação única*, uma classe de métodos de tratamento de dados faltantes a ser descrita no próximo capítulo, são os mais comumente utilizados para predizer tais informações inexistentes.

Capítulo 3

Métodos de tratamento de dados faltantes

Muitos métodos já foram propostos para tratar dados faltantes, mas poucos ganharam algum destaque (ALLISON, 2001). Este capítulo visa apresentar esses métodos que mereceram destaque, os quais são: caso completo, casos disponíveis, caso completo ponderado, imputação única, máxima verossimilhança e imputação múltipla.

3.1 Caso Completo

O método *Caso Completo*, também conhecido como *Listwise Deletion* (LD), descarta todos os objetos com algum atributo faltante, ou seja, somente são levados em consideração os casos completos (em que todos os atributos foram observados). Mesmo se houver apenas um atributo faltante, o objeto será descartado.

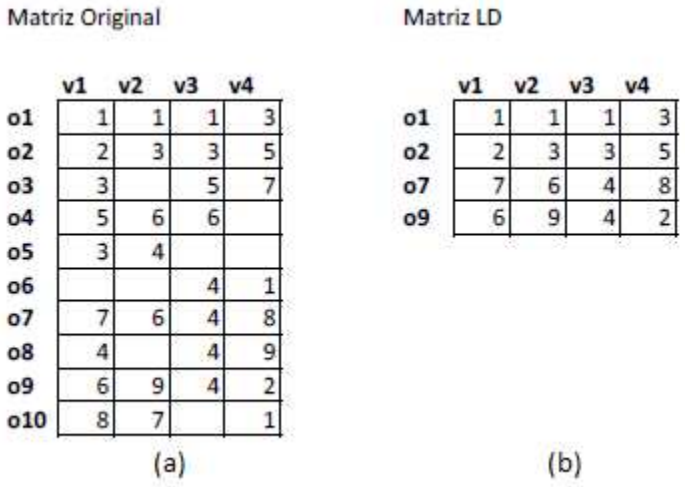
Este é um método que exige pouco esforço por parte do analista de dados, pois, considerando apenas os casos completos, forma-se uma base de dados completa, a qual pode ser analisada através de procedimentos analíticos convencionais (MCKNIGHT *et al.*, 2007). Nota-se, portanto, que o objetivo deste método não é estimar os valores faltantes, mas sim gerar uma matriz de dados que possa ser analisada através de procedimentos analíticos convencionais, a fim de se obter estimativas dos parâmetros de interesse.

A Figura 3.1.(a) traz um exemplo de uma matriz contendo 10 objetos e 4 variáveis, sendo que apenas 4 desses 10 objetos possuem todas as suas variáveis observadas. Desse modo, ao se aplicar o método LD, a matriz gerada conteria apenas esses 4 objetos, o que é mostrado na Figura 3.1.(b).

Este método possui três vantagens: (*i*) simplicidade, uma vez que as análises estatísticas convencionais podem ser aplicadas, sem nenhuma alteração (LITTLE & RUBIN, 2002; ALLISON, 2001); (*ii*) possibilidade de comparação das estatísticas univariadas, uma vez que todas as

variáveis possuem a mesma amostra (LITTLE & RUBIN, 2002); e (*iii*) não são necessários métodos computacionais especiais (ALLISON, 2001).

Apesar dessas vantagens, este método é aconselhável apenas quando os dados são MCAR, pois, segundo MCKNIGHT *et al.* (2007), "[...] se as estimativas dos parâmetros não são enviesadas para o conjunto de dados completo, também não serão para o subconjunto de dados formado após a aplicação deste método".



	v1	v2	v3	v4
o1	1	1	1	3
o2	2	3	3	5
o3	3		5	7
o4	5	6	6	
o5	3	4		
o6			4	1
o7	7	6	4	8
o8	4		4	9
o9	6	9	4	2
o10	8	7		1

	v1	v2	v3	v4
o1	1	1	1	3
o2	2	3	3	5
o7	7	6	4	8
o9	6	9	4	2

Figura 3.1: Matriz com dados faltantes (a) e matriz resultante do método LD (b).

As desvantagens desse método vêm da potencial perda de informação devido ao descarte dos casos incompletos. Esta perda de informação tem dois aspectos: perda de precisão e viés, quando o mecanismo não é MCAR (LITTLE & RUBIN, 2002), tornando este método uma opção indesejada. É mais provável que a perda de precisão e o viés sejam mínimos quando a porcentagem de casos completos for alta, mas isso não é uma regra, pois a perda de precisão e o viés também dependem dos parâmetros de interesse e do nível em que se diferem os casos completos dos incompletos (LITTLE & RUBIN, 2002).

3.2 Casos Disponíveis

O método de *Casos Disponíveis*, também chamado de *Pairwise Deletion* (PD), é bastante parecido com o LD. A diferença é que o PD não descarta os dados em nível de observação

(objeto), mas sim em nível de atributo(s) de interesse. Este é um procedimento alternativo para análises univariadas, pois ele inclui todos os objetos em que a variável de interesse está presente (LITTLE & RUBIN, 2002). A Figura 3.2 traz o exemplo de uma matriz contendo 10 objetos e 4 variáveis (a) e a matriz gerada através do procedimento PD (b), sendo *v4* a variável de interesse.

Como este método descarta os objetos em nível de variável e não em nível de observação, uma de suas desvantagens é que a amostra-base muda de variável para variável, de acordo com o padrão dos dados faltantes. Pode-se presumir que, para grandes bases de dados com diversos padrões de dados faltantes, os casos que provêem dados para uma variável podem ser completamente diferentes dos casos que provêem dados para outra variável. Isso implica em vários problemas no momento de: (*i*) analisar a matriz de covariância ou a de correlação, as quais são normalmente singulares ou indeterminadas (MCKNIGHT *et al.*, 2007); e (*ii*) calcular os *erros padrão* ou qualquer outra medida de incerteza (SCHAFER & GRAHAM, 2002). O erro padrão é uma medida das diferenças entre os valores amostrais observados e os valores preditos, que são obtidos com o uso da reta de regressão. Assim como o desvio-padrão é uma medida de como os valores se afastam de sua média, o erro padrão é uma medida de como os pontos amostrais se afastam de sua reta de regressão (TRIOLA, 2005). Também existe o erro padrão da média que analisa a variabilidade da média para diferentes amostras de mesmo tamanho.

Matriz Original

	v1	v2	v3	v4
o1	1	1	1	3
o2	2	3	3	5
o3	3		5	7
o4	5	6	6	
o5	3	4		
o6			4	1
o7	7	6	4	8
o8	4		4	9
o9	6	9	4	2
o10	8	7		1

(a)

Matriz PD

	v1	v2	v3	v4
o1	1	1	1	3
o2	2	3	3	5
o3	3		5	7
o6			4	1
o7	7	6	4	8
o8	4		4	9
o9	6	9	4	2
o10	8	7		1

(b)

Figura 3.2: Matriz com dados faltantes (a) e matriz resultante do processo PD (b), sendo *v4* a variável de interesse.

Segundo ALLISON (2001), se os dados são MCAR, PD possibilita estimativas consistentes dos parâmetros de interesse. Caso contrário, as estimativas podem ser seriamente enviesadas. LITTLE & RUBIN (2002) ressaltam que, para o caso do mecanismo ser MCAR, amostras-bases diferentes são aceitáveis para estimativas de média e variância, mas não para estimativas de covariância e correlação.

Através de simulações, KIM & CURRY (1977) (apud LITTLE & RUBIN, 2002) concluíram que o método PD é mais eficiente que o método LD quando o mecanismo é MCAR e as correlações são modestas. Porém, de um modo geral, nenhum desses dois métodos produz resultados satisfatórios (LITTLE & RUBIN, 2002).

3.3 Caso Completo Ponderado

O método *Caso Completo Ponderado* é uma extensão do método Caso Completo. O método Caso Completo Ponderado se baseia nos dados observados para associar um valor (chamado peso) aos casos completos, com o intuito de ajustar o viés (LITTLE & RUBIN, 2002). Ele tem o intuito de amenizar os problemas oriundos do método LD (tamanho menor da amostra e diminuição do poder estatístico na análise dos dados) (MCKNIGHT *et al.*, 2007).

Este é um método chamado de *método de ajuste*, porque o objetivo dos pesos é aproximar a distribuição dos casos completos da distribuição da amostra completa da população. Segundo MCKNIGHT *et al.* (2007), os pesos são empregados para corrigir a variabilidade da população e os erros padrão associados aos parâmetros. Através dos pesos, este método pode reduzir significativamente o viés associado às diferentes respostas possíveis, usadas para modelar as probabilidades de resposta. Mas este método não é capaz de corrigir o viés relacionado às variáveis não-usadas ou não-medidas (SCHAFER & GRAHAM, 2002).

Buscando produzir pesos adequados, para cada variável com dados faltantes é necessário que, a partir dos dados observados, sejam estimadas as probabilidades de cada possível resposta acontecer. Desse modo, este método pode ser enfadonho quando existem muitas variáveis com dados faltantes e que possuem diferentes probabilidades de resposta e/ou quando existem muitos padrões de dados faltantes (MCKNIGHT *et al.*, 2007). Como este cenário é comum, este método é indicado somente para algumas condições: quando existem poucos padrões de dados faltantes e

quando as probabilidades das respostas são conhecidas e relativamente uniformes entre as variáveis (MCKNIGHT *et al.*, 2007).

LITTLE & RUBIN (2002) afirmam que, como este método também desconsidera os casos incompletos e não provê controle interno da variância, ele é mais recomendado quando a informação sobre a covariância é limitada e o tamanho da amostra é grande, ou seja, quando o viés é um problema mais sério que a variância.

3.4 Imputação Única

Imputação é um termo genérico para o preenchimento de dados faltantes com valores plausíveis (SCHAFER, 1997). Os métodos de imputação podem ser de imputação única (IU) ou de imputação múltipla. Nesta seção, os procedimentos de imputação única serão abordados. A imputação múltipla será abordada na Seção 3.6.

O princípio básico dos métodos de IU é imputar um valor para cada dado faltante da base de dados e, então, analisá-la como se não houvesse dados faltantes (MCKNIGHT *et al.*, 2007). Desse modo, são muitos os métodos que podem receber esta denominação e, a seguir, serão apresentados alguns deles. Também nota-se que, ao contrário dos métodos apresentados nas seções anteriores, o objetivo deste método é predizer os valores faltantes. Como os sistemas de recomendação objetivam, basicamente, predizer a informação faltante, esses métodos se mostram adequados para guiar o processo de sugestão. As estimativas dos parâmetros de interesse ficam em segundo plano, mas podem ser facilmente calculadas, visto que os métodos de IU produzem bases de dados completas, as quais podem ser analisadas através de procedimentos analíticos convencionais.

3.4.1 Imputação por constantes

Dentre os métodos de IU, os métodos de imputação por constantes são os mais comuns (MCKNIGHT *et al.*, 2007). De forma geral, esses métodos substituem todos os valores faltantes de uma variável por um único valor (uma constante). Como exemplos desses métodos, é possível citar: imputação de zeros, imputação da média e imputação da mediana.

O método de imputação de zeros é a técnica mais simples entre os exemplos citados. Simplesmente consiste em trocar os valores faltantes por zero. Logicamente, isso só é possível se

zero for um valor plausível para o conjunto de dados. Se o motivo dos dados faltantes está ligado com maus resultados e os maus resultados estão associados a valores próximos de zero, a imputação de zeros pode não ter qualquer influência nos resultados da análise (MCKNIGHT *et al.*, 2007). Entretanto, como já foi dito, existem diversas razões que levam aos dados faltantes, e a grande maioria delas não justifica que a pior (ou mais conservadora) resposta seja considerada. Portanto, de modo geral, esta técnica não produz bons resultados (MCKNIGHT *et al.*, 2007). Além do mais, quanto maior a quantidade de dados faltantes, mais os resultados tendem a ser piores, devido à maior quantidade de zeros imputados.

A imputação da média é um método comum (MYRTVEIT *et al*., 2001) e bastante utilizado (BROWN & KROS, 2003), devido à sua facilidade de implementação. Nesta técnica, a média dos valores de um atributo que contém dados faltantes é usada para preencher os seus dados faltantes (no caso de atributos categóricos, a moda é usada no lugar da média) (FARHANGFAR *et al*., 2004). Apesar da fácil implementação, na maioria dos casos não é um método eficaz para o tratamento dos dados faltantes. Com a imputação da média, os valores extremos ficam sub-representados, implicando em perda de variabilidade. Portanto, a variância das variáveis com dados faltantes é subestimada (MCKNIGHT *et al.*, 2007). O efeito prejudicial deste método é reduzido somente em bases de dados com uma pequena porcentagem de dados faltantes (MCKNIGHT *et al.*, 2007). Também é importante enfatizar que a média é a melhor medida de tendência central para variáveis normalmente distribuídas. Desse modo, quando a distribuição normal não se verifica, os resultados deste método podem ser bastante pobres.

Basicamente, a imputação da mediana é bem parecida com a imputação da média, apresentando as mesmas vantagens e desvantagens. Porém, a imputação da mediana é uma alternativa melhor para variáveis que não são normalmente distribuídas, pois a mediana representa melhor a tendência central de uma distribuição que possui grandes desvios da distribuição normal.

3.4.2 Hot Deck e Cold Deck

As técnicas baseadas em Hot Deck (HD) preenchem um valor faltante de um atributo de um objeto a partir do(s) valor(es) observado(s) para este mesmo atributo em outro(s) objetos(s) do mesmo conjunto de dados (SCHAFER & GRAHAM, 2002).

Existem vários métodos HD e o mais simples consiste em imputar dados a partir de um objeto escolhido aleatoriamente (MCKNIGHT *et al.*, 2007). Outro método simplista é escolher um objeto com base em alguma informação, por exemplo, se uma informação faltante está num objeto que se refere a uma pessoa do sexo feminino, então algum objeto será escolhido aleatoriamente dentre os objetos que se referem às pessoas do sexo feminino para fornecer a informação a ser imputada (MCKNIGHT *et al.*, 2007).

Em um método mais elaborado de HD, para todo objeto que contém dados faltantes, o(s) objeto(s) mais semelhante(s) é(são) encontrado(s) e, então, os dados faltantes são imputados a partir desse(s) objeto(s) (LAKSHMINARAYAN *et al.*, 1999; FARHANGFAR *et al.*, 2004; BROWN & KROS, 2003). Logicamente, se um objeto dito similar também contém informações faltantes para os mesmos atributos, então ele é descartado, e outro objeto similar é procurado (FARHANGFAR *et al.*, 2004; FARHANGFAR *et al.*, 2007).

As vantagens dos métodos HD são: (*i*) simplicidade conceitual; (*ii*) bom nível de medição de variáveis; e (*iii*) como em todos os métodos de imputação, uma base de dados completa é gerada, a qual pode ser analisada através de procedimentos analíticos convencionais (BROWN & KROS, 2003).

Uma das desvantagens de HD é a dificuldade em definir o que é *similar*. Embora existam várias métricas que podem ser utilizadas, como a distância euclidiana e a correlação de Pearson, a escolha da métrica certamente vai influenciar no desempenho de HD (BROWN & KROS, 2003).

Outra desvantagem é a subestimação dos erros padrão, devido à diminuição da variabilidade nas variáveis com dados faltantes, porque tendem a serem imputados valores distantes dos extremos da distribuição (MCKNIGHT *et al.*, 2007). O problema de subestimar os erros padrão é que isso pode implicar em erros do *Tipo I* (MCKNIGHT *et al.*, 2007). Erros do *Tipo I* significam que se pode aceitar uma afirmação sem evidências suficientes.

Os métodos Cold Deck (CD) são muito similares aos métodos HD. Basicamente, o que os difere é que no HD os dados utilizados para a substituição dos dados faltantes estão no próprio conjunto de dados. Enquanto isso, no CD, estão em outro conjunto de dados (LAKSHMINARAYAN *et al.*, 1999; ACUNA & RODRIGUEZ, 2004). Por exemplo, numa pesquisa longitudinal, essa fonte de dados pode ser a medição anterior ou a posterior. É importante se certificar de que a fonte de dados externa contenha valores factíveis para o preenchimento dos dados faltantes (BROWN &

KROS, 2003). Através dessa fonte de dados externa, CD procura diminuir o problema da perda de variabilidade, visando assim evitar os erros do *Tipo I* (MCKNIGHT *et al.*, 2007).

3.4.3 Outras técnicas de imputação única

Além das técnicas de IU já citadas, existem diversas outras, como por exemplo: Imputação de Médias Condicionadas (CMI – do inglês *Conditional Mean Imputation*), Próximo Valor Carregado para Trás (NVCB – do inglês *Next Value Carried Backward*) e Último Valor Carregado para Frente (LVCF – do inglês *Last Value Carried Forward*) (MCKNIGHT *et al.*, 2007).

Ao contrário do método de imputação de médias, que para calcular a média de uma variável utiliza todos os registros observados desta variável, o método CMI calcula a média para diferentes subgrupos formados a partir de variáveis de classificação (MCKNIGHT *et al.*, 2007). Por exemplo, se *sexo* é a única variável de classificação e tem-se um dado faltante para uma pessoa do sexo feminino, este será substituído pela média calculada dentro do grupo de pessoas do sexo feminino. Logicamente, quanto mais fraca a relação entre as variáveis de classificação e a variável com dados faltantes, mais este método se aproxima de um método de imputação de valores aleatórios (MCKNIGHT *et al.*, 2007).

Em estudos longitudinais, é possível utilizar os métodos LVCF ou NVCB. O método LVCF consiste basicamente em substituir o valor faltante de uma variável para um objeto pelo valor dessa mesma variável na medição anterior desse objeto. Já o método NVCB consiste basicamente em substituir o valor faltante de uma variável para um objeto pelo valor dessa variável na medição posterior desse objeto.

3.5 Máxima Verossimilhança

Os métodos de Máxima Verossimilhança (ML – do inglês *Maximum Likelihood*) são baseados em modelo. Isso significa que as estimativas dos parâmetros são calculadas a partir dos dados observados, das relações existentes entre os atributos observados e das restrições impostas pela suposição do modelo de distribuição (MCKNIGHT *et al.*, 2007).

Diferentemente dos métodos de imputação, o objetivo principal não é preencher os valores faltantes, mas sim estimar os parâmetros de interesse. Os métodos baseados em modelo tratam os

dados como se todos os valores tivessem sido observados, pois eles combinam os dados com um modelo teórico (por exemplo, a distribuição normal multivariada), e assim são estimados os parâmetros de interesse (MCKNIGHT *et al.*, 2007).

O princípio básico dos métodos ML é escolher como estimativa dos parâmetros aqueles valores que, se verdadeiros, maximizariam a probabilidade de observar o que, de fato, foi observado (ALLISON, 2001). Dessa forma, ML é um método para estimar parâmetros em uma variedade de modelos, como o *Modelo Linear Hierárquico* e o *Modelo Estrutural de Equações* (MCKNIGHT *et al.*, 2007). Um exemplo de procedimento ML é o critério dos *quadrados mínimos*, o qual considera que os erros são normalmente distribuídos, e as estimativas dos parâmetros são derivadas em conformidade com esta restrição (MCKNIGHT *et al.*, 2007).

Para iniciar um procedimento ML, é considerado que os dados são gerados por um modelo descrito pela função de densidade $f(A|\theta)$, onde A são os dados e θ é um conjunto de parâmetros desconhecidos que governa a distribuição de A, do qual sabe-se apenas estar situado no espaço Ω_θ (LITTLE & RUBIN, 2002). Salvo indicação contrária, consideram-se intervalos adequados para os elementos de θ, por exemplo: o espaço dos reais para médias, os reais positivos para variâncias e o intervalo [0,1] para probabilidades (LITTLE & RUBIN, 2002). Portanto, dado o modelo considerado e uma vez calculado o vetor de parâmetros θ, $f(A|\theta)$ pode ser empregado para amostrar valores faltantes (LITTLE & RUBIN, 2002; ALLISON, 2001).

A função de verossimilhança $L(\theta|A)$ é uma função do vetor de parâmetros $\theta \in \Omega_\theta$ dado A, proporcional à função de densidade; por definição, $L(\theta|A) = 0$ para qualquer $\theta \notin \Omega_\theta$ (LITTLE & RUBIN, 2002). Note que a função de verossimilhança é uma função do conjunto de parâmetros θ para A fixo, enquanto a função de probabilidade ou densidade é uma função de A para θ fixo. Também nota-se que não existe somente uma única função de verossimilhança, visto que ela consiste num conjunto de funções que diferem entre si por qualquer fator que não dependa de θ (LITTLE & RUBIN, 2002).

Quando as observações são independentes (que é a suposição usual), a verossimilhança global para uma amostra é dada pelo produto de todas as verossimilhanças para as observações individuais (ALLISON, 2001). Suponha que exista uma variável Y no conjunto de dados A e que se deseja estimar o parâmetro θ. Se $f(y|\theta)$ é a probabilidade (ou densidade de probabilidade) de

observar um único valor de *Y* dado algum valor de θ, a verossimilhança para uma amostra de *n* observações é (ALLISON, 2001):

$$L(\theta \mid Y) = \prod_{i=1}^{n} f(y_i \mid \theta) \tag{3.1}$$

Segundo LITTLE & RUBIN (2002), em alguns problemas é, de certa forma, mais conveniente trabalhar com a função $l(\theta|A)$ (*log-verossimilhança*), que é o logaritmo natural (ln) da função de verossimilhança. Como o logaritmo é uma função monotonicamente crescente, o logaritmo da função alcança o seu valor máximo nos mesmos pontos que a função original. Logo, a função $l(\theta|A)$ pode ser usada no lugar de $L(\theta|A)$ na estimação da máxima verossimilhança. Encontrar o máximo de uma função sempre envolve o cálculo de derivadas e isso se torna mais fácil quando a função que está sendo maximizada é $l(\theta|A)$.

O objetivo dos métodos de ML é encontrar o valor de $\theta \in \Omega_\theta$ que maximiza a função de verossimilhança $L(\theta|A)$, ou, equivalentemente, $l(\theta|A)$. É possível encontrar mais de uma estimativa ML. Porém, para muitos modelos importantes, a estimativa ML é única (LITTLE & RUBIN, 2002).

Segundo ALLISON (2001), quando o mecanismo dos dados faltantes é ignorável, pode-se obter a verossimilhança simplesmente pela soma da verossimilhança usual sobre todos os possíveis valores dos dados faltantes. Suponha, por exemplo, que se tentou coletar dados em duas variáveis, *X* e *Y*, para uma amostra independente de *n* observações. Para as primeiras *m* observações, ambas as variáveis foram observadas, mas para as *n* – *m* observações restantes, somente *Y* foi observada. Para uma única observação com dados completos, a função de densidade é descrita por $f(x, y|\theta)$, onde θ é um conjunto de parâmetros desconhecidos que governam a distribuição de *X* e *Y*. Supondo que *X* é discreto, a função de densidade para um caso com dados faltantes em *X* é a distribuição marginal de *Y*:

$$g(y \mid \theta) = \sum_{x} f(x, y \mid \theta). \tag{3.2}$$

Logo, a verossimilhança para toda a amostra é:

$$L(\theta \mid X, Y) = \prod_{i=1}^{m} f(x_i, y_i \mid \theta) \prod_{i=m+1}^{n} g(y_i \mid \theta). \tag{3.3}$$

Para exemplificar essas características de ML, a Subseção 3.5.1 traz um exemplo baseado em ALLISON (2001).

As técnicas ML são conhecidas na literatura pela sua eficiência e estão implementadas em vários softwares estatísticos (MYRTVEIT *et al.*, 2001). Estes métodos possuem várias características desejáveis (MCKNIGHT *et al.*, 2007; ALLISON, 2001). A primeira delas é produzir estimativas aproximadamente não-enviesadas para grandes amostras. A segunda delas é que os erros padrão são, ao menos, tão pequenos quanto os erros padrão produzidos por outro método consistente. E, por fim, em amostragens repetidas, as estimativas têm aproximadamente uma distribuição normal, o que pode ser utilizado para calcular intervalos de confiança. Vale ressaltar que este método é considerado eficiente sob o mecanismo ignorável e também que, quanto maior a amostra, maior a tendência de se gerar bons resultados.

Como os métodos ML requerem que seja considerado um modelo, é de extrema importância que esta escolha seja bem feita. Caso contrário, estes métodos podem sofrer uma diminuição drástica de precisão. Logo, este é o ponto fraco deste método, pois nem sempre o analista de dados vai conseguir propor um modelo adequado (ALLISON, 2001).

3.5.1 Exemplo de um procedimento simples de ML

Suponha que, para uma amostra aleatória simples de 500 pessoas, tentou-se medir 2 variáveis dicotômicas, ou seja, variáveis que podem assumir um dentre dois valores distintos. Sejam essas variáveis X e Y. Para 375 pessoas, conseguiu-se medir X e Y, os resultados obtidos são mostrados na Tabela 3.1.

Tabela 3.1: Tabela de Contingência.

	$Y = 1$	$Y = 2$
$X = 1$	130	53
$X = 2$	85	107

Para os outros 125 casos, X é faltante e apenas Y foi observado. Especificamente, têm-se 48 casos com $Y = 1$ e 77 casos com $Y = 2$. Na população, o relacionamento entre X e Y é descrito segundo a Tabela 3.2, na qual p_{ij} é a probabilidade de $X = i$ e $Y = j$. Se houvesse somente os 375 casos completos, a função de verossimilhança seria:

$$L = (p_{11})^{130}(p_{12})^{53}(p_{21})^{85}(p_{22})^{107},$$

lembrando que a soma das 4 probabilidades deve ser igual a 1. As estimativas ML para as 4 probabilidades poderiam ser, simplesmente, a proporção em cada célula da Tabela 3.1:

$$\hat{p}_{ij} = \frac{n_{ij}}{n},$$

onde n_{ij} é o número de casos presentes na célula (i, j) e n é a quantidade de casos completos, ou seja, 375. Calculando, obtêm-se:

$$\hat{p}_{11} = 0{,}347,$$
$$\hat{p}_{12} = 0{,}141,$$
$$\hat{p}_{21} = 0{,}227,$$
$$\hat{p}_{22} = 0{,}285.$$

Tabela 3.2: Tabela de probabilidades.

	$Y = 1$	$Y = 2$
$X = 1$	p_{11}	p_{12}
$X = 2$	p_{21}	p_{22}

Entretanto, também é desejável levar em consideração as informações adicionais que existem em Y somente, as quais precisam ser incorporadas na função de verossimilhança. Considerando que o mecanismo é ignorável, a verossimilhança para casos com $Y = 1$ é $p_{11} + p_{21}$, isto é, a probabilidade marginal de $Y = 1$. Similarmente, para casos com $Y = 2$, a verossimilhança é $p_{12} + p_{22}$. Assim, a função de verossimilhança para a amostra toda é:

$$L = (p_{11})^{130}(p_{12})^{53}(p_{21})^{85}(p_{22})^{107}(p_{11} + p_{21})^{48}(p_{12} + p_{22})^{77}.$$

Para a maioria das aplicações ML para problemas de dados faltantes, não existe uma solução explícita para as estimativas de p_{ij} que maximizariam essa expressão. Em vez disso, são necessários métodos iterativos. Nesse exemplo, entretanto, o padrão dos dados faltantes é necessariamente monotônico (porque só existe uma variável com dados faltantes), então pode-se estimar a distribuição condicional de X dado Y separadamente. Logo, as estimativas ML têm a forma geral:

$$\hat{p}_{ij} = \hat{p}(X = i \,|\, Y = j)\hat{p}(Y = j).$$

As probabilidades condicionais do lado esquerdo são estimadas usando somente os casos completos, de maneira usual. Considerando a Tabela 3.1, basta dividir a frequência de cada célula pelo total de sua respectiva coluna. As estimativas da probabilidade marginal de Y são obtidas através da soma da frequência da coluna $Y = j$ da Tabela 3.1 mais a frequência de $Y = j$ para os casos com dados faltantes em X e, então, realizando a divisão pelo tamanho da amostra. Assim, as probabilidades p_{ij} são dadas por:

$$\hat{p}_{11} = \left(\frac{130}{215}\right)\left(\frac{215+48}{500}\right) = 0{,}318,$$
$$\hat{p}_{12} = \left(\frac{53}{160}\right)\left(\frac{160+77}{500}\right) = 0{,}157,$$
$$\hat{p}_{21} = \left(\frac{85}{215}\right)\left(\frac{215+48}{500}\right) = 0{,}208,$$
$$\hat{p}_{22} = \left(\frac{107}{160}\right)\left(\frac{160+77}{500}\right) = 0{,}317.$$

Como já foi dito, para casos mais complexos, com várias variáveis e/ou com o padrão dos dados faltantes diferente de monotônico, são necessários procedimentos iterativos. Para exemplificar, uma das técnicas existentes é descrita na Subseção 3.5.2.

3.5.2 O algoritmo EM

O algoritmo EM é um método geral para obter estimativas ML em bases de dados incompletas (LITTLE & RUBIN, 2002). Como as estimativas ML podem ser difíceis de obter para bases de dados complexas ou do "mundo real", é necessário um procedimento para reduzir esta dificuldade, e é este o intuito do algoritmo EM (MCKNIGHT *et al.*, 2007).

O nome *EM* vem de seus dois principais passos: Esperança e Maximização. Este é um algoritmo iterativo, pois estes dois passos se repetem até a convergência. Sucintamente, pode-se explicar o algoritmo da seguinte maneira:

- Escolher valores iniciais para os *parâmetros* do modelo considerado (por exemplo, médias e matriz de covariância para o modelo normal multivariado). Estes valores iniciais podem ser obtidos através das fórmulas convencionais, usando *Listwise Deletion* ou *Pairwise Deletion* (ALLISON, 2001).
- Faça até a convergência:

- *Esperança*: imputar valores para os dados faltantes baseando-se nos valores dos *parâmetros*. Para exemplificar, considere que em uma análise de dados trabalha-se com a distribuição normal multivariada, que a base de dados possui 4 atributos (A_1, A_2, A_3, A_4) e que exista alguns dados faltantes em A_3 e A_4 (sendo que o padrão de dados faltantes é arbitrário). Os valores podem ser imputados através da regressão de A_3 em A_1 e A_2 e da regressão de A_4 em A_1 e A_2 (ALLISON, 2001).
- *Maximização*: estimar novos valores dos *parâmetros*. Retornando ao exemplo do item anterior, a média pode ser calculada utilizando a fórmula convencional, mas a variância e a covariância devem levar em consideração um termo adicional que corresponde aos resíduos da variância e covariância (ALLISON, 2001). Por exemplo, suponha que para a observação *i*, A_3 foi imputado usando A_1 e A_2. Logo, para o cálculo da variância deve ser utilizado $(a_{i3})^2 + s^2_{3.21}$, onde $(a_{i3})^2$ faz parte da fórmula convencional da variância e $s^2_{3.21}$ é o resíduo da variância da regressão de A_3 em A_1 e A_2. A adição do termo residual corrige a subestimação da variância que ocorre nos esquemas mais convencionais de imputação (ALLISON, 2001).

O método converge quando a diferença entre os valores estimados dos parâmetros em duas iterações consecutivas é menor que um limiar pré-estabelecido.

Como o algoritmo EM é um método baseado em ML, ele possui propriedades desejáveis quando o mecanismo dos dados faltantes for ignorável, produzindo estimativas ML ótimas (isto é, consistentes e eficientes) para grandes amostras. Entretanto, os procedimentos baseados em modelos oferecem pouca ajuda na área de testes de hipóteses. Mais especificamente, o procedimento EM tende a subestimar os erros padrão, que são críticos para os testes de hipóteses, podendo gerar erros do *Tipo I* (MCKNIGHT *et al.*, 2007; GRAHAM, 2009).

Pode haver dois inconvenientes maiores com o algoritmo EM: (*i*) em alguns casos, com grandes frações de informações faltantes, ele pode ser muito lento para convergir; e (*ii*) em algumas aplicações, o passo de maximização não tem uma formulação computacional simples (LITTLE & RUBIN, 2002). Existem algumas extensões do algoritmo EM que tentam amenizar esses inconvenientes, como por exemplo, o algoritmo ECM (MENG & RUBIN, 1993).

O algoritmo EM também não garante a convergência para o ótimo global se a função de verossimilhança (ou log-verossimilhança) for multimodal, possuindo ótimos locais. Logo, uma boa escolha dos valores iniciais dos parâmetros é importante para alcançar o ótimo global.

3.6 Imputação Múltipla

Os métodos de imputação múltipla (MI – do inglês *Multiple Imputation*) foram propostos por RUBIN (1987). Surgiram como uma alternativa flexível aos métodos ML para uma grande variedade de problemas de dados faltantes (SCHAFER & GRAHAM, 2002). Esse método (*i*) provê estimativas de parâmetros confiáveis (incluindo erros padrão); (*ii*) permite a estimação da informação faltante e o seu impacto nas estimativas dos parâmetros; e (*iii*) pode ser aplicado em diferentes situações de dados faltantes (MCKNIGHT *et al.*, 2007).

Enquanto os métodos de imputação única substituem cada valor faltante por um único valor, estes métodos substituem por x valores, com $x \geq 2$. Desse modo, são formadas x bases de dados completas, que, portanto, podem ser analisadas através de procedimentos convencionais. Os resultados dessas análises são agregados e, por fim, a informação faltante pode ser computada. Em resumo, é esse o procedimento de um método de MI. Em mais detalhes, os passos de MI serão listados na Subseção 3.6.1.

Como MI é o nome dado a uma família de métodos, é necessário saber selecionar o método adequado. O processo de seleção do método MI adequado deve ser guiado pela suposição sobre a distribuição dos dados, como também pelos tipos de dados (contínuos, discretos e/ou categóricos) presentes na base de dados (LAKSHMINARAYAN *et al.*, 1999). Isso é necessário, porque em procedimentos MI um valor faltante é substituído com base nos valores observados e em um erro que é adicionado para garantir a conformidade com a distribuição considerada (MCKNIGHT *et al.*, 2007).

A distribuição mais comumente utilizada é a normal (ALLISON, 2001), mas, mesmo quando algumas variáveis não são normais, esse modelo pode ser utilizado (SCHAFER, 1997). Além do mais, quando os dados de uma variável não são normais, muitas vezes é possível transformá-los para que eles respeitem uma distribuição normal.

3.6.1 Passo a passo da MI

A Figura 3.3 exibe os 4 passos da técnica de MI: imputação, análise das bases de dados geradas, agregação dos resultados e cálculo da informação faltante. Esses passos são explicados com mais detalhes nas subseções a seguir.

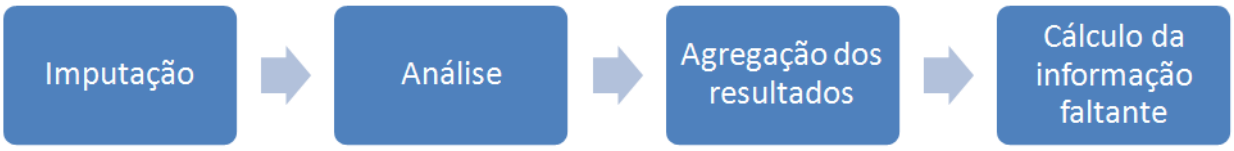


Figura 3.3: Os 4 passos da técnica de MI.

3.6.1.1 Imputação

Este é o passo fundamental da técnica de MI (ZHANG, 2003). O passo de imputação é semelhante à IU, porém são realizadas $x \geq 2$ imputações para cada valor faltante. Não existem restrições quanto ao método escolhido para realizar as x imputações, os quais podem ser, por exemplo, estimativas ML, Hot Deck ou Cold Deck. Entretanto, utilizar mais de um método não é, geralmente, sensato, porque eles produziriam resultados um pouco diferentes. Essas diferenças influenciariam na estimativa da informação faltante, produzindo resultados piores devido ao aumento da variabilidade. A maioria dos especialistas em MI recomenda um procedimento iterativo que não é limitado somente a um grupo específico de valores imputados (como a média, por exemplo). Ao invés disso, um processo aleatório é preferível, de modo que os valores sejam únicos em cada conjunto imputado, mas compartilhem uma relação comum subjacente aos dados (MCKNIGHT *et al.*, 2007). Essa variação aleatória é introduzida para evitar que as variâncias das variáveis sejam subestimadas, as quais são críticas para as inferências estatísticas (ALLISON, 2001; MCKNIGHT *et al.*, 2007).

Enquanto a variação aleatória introduzida nas imputações pode eliminar o viés que é endêmico às imputações determinísticas, as imputações múltiplas podem ajustar os erros padrão para cima, reduzindo a probabilidade de erros do *Tipo I* (ALLISON, 2001; MCKNIGHT *et al.*, 2007; SCHAFER & GRAHAM, 2002). Isso acontece devido ao componente aleatório, que faz com que as estimativas dos parâmetros de interesse sejam levemente diferentes em cada base de dados imputada.

Também é necessário frisar que qualquer técnica utilizada para tratar os dados faltantes é influenciada pelo tipo de dados envolvido (LAKSHMINARAYAN *et al.*, 1999). Desse modo, a escolha do método de imputação deve levar em consideração os tipos de dados presentes na base.

3.6.1.2 Análise

As $x \geq 2$ bases de dados geradas são usadas em análises convencionais para testar os modelos estatísticos de interesse para o estudo. Cada uma das x bases de dados completas é analisada individualmente. Logicamente, como são realizadas x análises, serão geradas x estimativas de cada um dos parâmetros de interesse.

Não existe nenhuma restrição quanto ao tipo de análise estatística, de modo que análises univariadas ou multivariadas são igualmente aplicáveis (MCKNIGHT *et al.*, 2007).

Por exemplo, suponha que exista uma base de dados com quatro variáveis: preço, peso, cor e claridade de um diamante, sendo que o preço do diamante é a variável dependente. Suponha também que a regressão múltipla seja o modelo estatístico de interesse para o estudo em questão. Então, após realizar a imputação, é possível analisar cada uma das bases de dados através da regressão múltipla. Desse modo, para cada base de dados completa, pode-se encontrar o coeficiente de regressão de cada uma das variáveis independentes e também o intercepto, além dos erros padrão. Nesse exemplo, os coeficientes de regressão e o intercepto são os parâmetros de interesse.

3.6.1.3 Agregação dos resultados

O terceiro passo é agregar os resultados das análises feitas nas $x \geq 2$ bases de dados, gerando estimativas globais para os parâmetros de interesse e para o erro padrão.

Uma maneira simples para se gerar uma estimativa global para um parâmetro de interesse Q é através da média das estimativas produzidas para as x bases de dados (MCKNIGHT *et al.*, 2007). Para o exemplo dado anteriormente, poderia ser calculada a média dos valores dos coeficientes de regressão. Cada parâmetro estimado é chamado de $\hat{Q}$ e a estimativa global é chamada de $\overline{Q}$.

Para calcular o erro padrão global, que é necessário para os testes de significância e para os intervalos de confiança (MCKNIGHT *et al.*, 2007), são necessários vários passos (RUBIN, 1987).

Primeiramente, é necessário calcular a média dos erros padrão (que foram calculados através dos x conjuntos de dados completos) ao quadrado, o que é chamado de *within-imputation variance*:

$$\overline{U} = \frac{1}{x}\sum_{j=1}^{x} U_j, \tag{3.4}$$

onde U_j é o erro padrão ao quadrado calculado através do conjunto de dados completo *j*. E também é necessário calcular a variância do parâmetro de interesse estimado, o que é chamado de *between-imputation variance*:

$$B = \frac{1}{x-1}\sum_{j=1}^{x} (\hat{Q}_j - \overline{Q})^2. \tag{3.5}$$

Assim, é possível calcular a *variância total* (*T*). Tipicamente, $T = \overline{U} + B,$ mas a abordagem de RUBIN (1987) foi ponderar *B* de acordo com o número de imputações realizadas. Assim, a variância total de $\left(\hat{Q}_j - \overline{Q}\right)$ é dada por:

$$T = \overline{U} + \left(1 + \frac{1}{x}\right)B. \tag{3.6}$$

O erro padrão global pode finalmente ser calculado como $\sqrt{T}$. Como já foi dito, o erro padrão global é necessário para se calcular intervalos de confiança e níveis de significância de *Q*, os quais são construídos utilizando uma distribuição de referência *t* com

$$df = (x-1)\left(1 + r^{-1}\right)^2 \tag{3.7}$$

graus de liberdade (*df* – do inglês *degrees of freedom*), onde *r* representa o aumento relativo na variância devido aos dados faltantes e é dado por:

$$r = \frac{\left(1 + x^{-1}\right)B}{\overline{U}}. \tag{3.8}$$

Quanto maior o valor de *r*, menor a estabilidade nos parâmetros estimados, refletindo em menor certeza estatística (MCKNIGHT *et al.*, 2007).

Assim, de forma padrão, um intervalo com 100(1 – α)% de confiança de $\overline{Q}$ é:

$$\overline{Q} \pm t_{df}(\alpha/2)\sqrt{T}, \tag{3.9}$$

onde $t_{df}(\alpha/2)$ é um quantil da distribuição *t* de Student com *df* graus de liberdade. Por exemplo, se *df* = ∞ e 1 – α = 0,95, $t_{df}(\alpha/2) = 1{,}96$. Logo, para calcular um intervalo de confiança de 95% basta fazer:

$$\overline{Q} \pm 1{,}96(\sqrt{T})$$

Para calcular o *valor t*, similar ao teste *t* de Student, basta fazer:

$$t(df) = \frac{\overline{Q}}{\sqrt{T}}. \quad (3.10)$$

A comparação entre $t(df)$ e df com relação à distribuição t de Student permite testar a hipótese nula, em que $Q = 0$ (MCKNIGHT *et al.*, 2007).

Segundo SCHAFER & GRAHAM (2002), quando df é grande, a distribuição de t é essencialmente normal, a variância total é bem estimada e pouco se ganha em aumentar o valor de x (SCHAFER & GRAHAM, 2002).

3.6.1.4 Cálculo da informação faltante

Os métodos de MI são os únicos capazes de estimar a influência dos dados faltantes na estimação dos parâmetros de interesse. Ao calcular o valor de r, quanto maior a variabilidade nos parâmetros estimados (dado pelo numerador B) e maior a confiabilidade de cada parâmetro estimado (isto é, erros padrão menores, dado pelo denominador $\overline{U}$), maior o valor de r. O aumento relativo na variabilidade fornece uma pista do impacto dos dados faltantes nas estimativas dos parâmetros: menor estabilidade nas estimativas dos parâmetros reflete em menor certeza estatística (MCKNIGHT *et al.*, 2007). A *taxa de informação* (γ) faltante fornece ao pesquisador uma noção do impacto dos dados faltantes nas estimativas dos parâmetros e, ultimamente, nas conclusões estatísticas. Em essência, a taxa de informação faltante serve como um indicador da incerteza estatística devido aos dados faltantes. Isso é relevante, porque não necessariamente existe relação entre a quantidade de dados faltantes e o seu verdadeiro impacto nas análises estatísticas (MCKNIGHT *et al.*, 2007). A taxa de informação faltante pertence ao intervalo [0, 1], onde 1 significa que existe 100% de informação faltante. Logo, quanto maior o valor de γ, menor a certeza estatística:

$$\gamma = \left(r + \frac{2}{df + 3} \right) (r+1)^{-1}. \quad (3.11)$$

RUBIN (1987) também mostrou que a eficiência de uma estimativa baseada em $x \geq 2$ imputações em relação a uma baseada em um número infinito de imputações é:

$$Efic = \frac{1}{1 + \gamma / x}. \quad (3.12)$$

Essa habilidade de estimar a influência dos dados faltantes na estimação dos parâmetros é um diferencial dos métodos de MI. Percebe-se que a eficiência relativa da inferência MI está

relacionada com a taxa de informações faltantes (γ) em combinação com o número de imputações (x). Segundo RUBIN (1987), o valor de x pode, geralmente, ser restringido a um valor menor que 10. Apesar de testes realizados por outros autores (HERSHBERGER & FISHER, 2003) mostrarem que, em alguns casos, há a necessidade de um valor maior de x, é razoável considerar esta heurística de Rubin, utilizando x entre 3 e 10 (MCKNIGHT *et al.*, 2007).

3.6.2 Considerações Finais sobre a MI

Segundo LITTLE & RUBIN (2002), MI tem as mesmas vantagens de IU e está isenta de algumas desvantagens associadas aos métodos de IU, como por exemplo:

- quando as $x \geq 2$ imputações são realizadas utilizando o mesmo modelo, os resultados das análises das $x \geq 2$ bases de dados podem ser facilmente combinados para criar uma inferência que reflete adequadamente a variabilidade da amostra; e
- quando as MIs são feitas a partir de mais de um modelo, a incerteza sobre o modelo correto é mostrada pela variação nas inferências válidas entre os modelos.

Os métodos MI também estão isentos de algumas desvantagens dos métodos ML. Ao contrário de ML, MI pode ser utilizado com praticamente qualquer tipo de dados e qualquer tipo de modelo, e a análise pode ser feita com softwares convencionais, sem nenhuma modificação (ALLISON, 2001). Além disso, MI é provavelmente menos sensível do que ML à escolha do modelo porque o modelo é usado somente para a imputação dos valores e não para estimar os parâmetros (ALLISON, 2001).

Um dos principais inconvenientes é que são produzidas estimativas um pouco diferentes a cada vez que você utiliza MI. Isso se deve à variação aleatória que é introduzida no processo de imputação, para evitar a subestimação da variância e possivelmente da covariância das variáveis com dados faltantes (ALLISON, 2001).

Segundo ALLISON (2001), MI é uma técnica de implementação difícil. Porém, com o avanço da computação e com a proliferação dos softwares para realizar MI, ela se tornou rapidamente o método padrão para manipular dados faltantes (MCKNIGHT *et al.*, 2007). Um exemplo de software que trabalha com MI é o *NORM*, um programa gratuito para Windows, que cria MIs para bases de dados incompletas, com padrão arbitrário, sob um modelo normal não-estruturado (SCHAFER & GRAHAM, 2002). O NORM será explicado com maiores detalhes na

subseção subsequente, pois no Capítulo 7, os resultados obtidos pelo ele serão comparados aos resultados obtidos pela biclusterização aplicada ao método de MI.

É importante salientar que, apesar da MI oferecer várias vantagens com relação aos métodos de deleção, imputação única e baseados em modelo, ela não é a melhor opção sempre. Por exemplo, como a MI é um método de simulação, ela pode ser menos eficiente sob condições em que métodos baseados em modelo podem calcular os parâmetros de interesse diretamente do conjunto de dados incompleto (MCKNIGHT *et al.*, 2007).

3.6.3 NORM

O NORM é um programa para o sistema operacional Windows (SCHAFER, 1999). O seu nome se refere à *distribuição normal multivariada*, ou seja, se refere ao modelo de distribuição dos dados utilizado para realizar a MI. Além de considerar a distribuição normal conjunta para as variáveis da base de dados a ser analisada, o NORM também supõe que o mecanismo associado aos dados faltantes é ignorável.

Os principais procedimentos do NORM são:

- Um algoritmo EM para estimação eficiente das médias, variâncias e covariâncias (ou correlações).
- Um procedimento de *data augmentation* (DA) para gerar as imputações múltiplas dos dados faltantes.

DA é um tipo especial de método baseado em Monte Carlo em cadeias de Markov, ou seja, é uma técnica de simulação iterativa. Em DA existem três tipos de quantidades: os dados observados, os dados faltantes e os parâmetros. Os dados faltantes e os parâmetros são desconhecidos. DA é também um procedimento bayesiano, portanto depende de uma distribuição a priori para os parâmetros desconhecidos θ (médias e matriz de covariância). A opção padrão do NORM é utilizar uma distribuição a priori não-informativa. Então, a densidade a priori para θ é $|\Sigma|^{-(p+1)/2}$, onde Σ é a matriz de covariância e p é o número de variáveis no modelo (SCHAFER, 1999). Essa função é a densidade a priori não-informativa padrão para análises bayesianas envolvendo a distribuição normal multivariada.

O procedimento DA possui dois passos básicos que se repetem até a convergência:

- Passo *I*: imputar os dados faltantes através de sua distribuição condicional, considerando os dados observados e os valores atuais dos parâmetros.
- Passo *P*: simular novos valores para os parâmetros através da distribuição posterior bayesiana, considerando os dados observados e a valores imputados no passo *I*.

Alternando entre essas duas etapas, estabelece-se uma cadeia de Markov que converge para uma distribuição estacionária, a distribuição conjunta dos dados faltantes e dos parâmetros dado o conjunto dos dados observados.

Segundo SCHAFER (1999), o algoritmo EM pode ser utilizado não só para obter as estimativas iniciais dos parâmetros, mas também para estimar o número de iterações necessárias do procedimento DA. Ele afirma que esse valor deve ser igual ou superior ao número de iterações necessárias para o algoritmo EM convergir.

3.7 Síntese do Capítulo

Este capítulo descreveu alguns dos métodos mais conhecidos para o tratamento de dados faltantes, assim como as vantagens e desvantagens de cada um desses métodos.

Basicamente, os métodos apresentados podem ser divididos em dois grupos: (*i*) aqueles que almejam predizer os parâmetros de interesse; e (*ii*) aqueles que almejam predizer os valores dos dados faltantes.

Logicamente, através dos métodos do segundo grupo, também é possível estimar os parâmetros de interesse. Mas o que é realmente interessante é que esse grupo de métodos viabiliza muito mais que somente análises estatísticas da base de dados. Como exemplo, citam-se os sistemas de recomendação.

No próximo capítulo, uma técnica chamada biclusterização será apresentada. Trata-se de uma técnica capaz de extrair subconjuntos de linhas e colunas de uma matriz de dados, de modo que os elementos desses subconjuntos compartilhem alguma correlação entre si. Desse modo, a biclusterização pode extrair informações relevantes de uma base de dados, as quais podem ser utilizadas para a construção de métodos de imputação, como será visto no Capítulo 5.

Capítulo 4

Biclusterização

Existem muitas técnicas de análise de dados e a *clusterização* é uma delas. Esta técnica pode ser definida como a organização ou separação de uma coleção de padrões – os dados – em grupos (os chamados *clusters*), baseando-se na *similaridade* ou na *dissimilaridade* existente entre eles, de modo que os padrões dentro de um cluster sejam mais similares entre si do que são com os padrões fora do cluster (padrões que pertencem a outros clusters) (DE CASTRO, 2006).

Dada uma matriz A, com M linhas, que representam os objetos, e N colunas, que representam os atributos, a técnica de clusterização irá agrupar estes objetos em c *clusters*, criados com base nos N atributos. Este procedimento, baseado em todos os atributos para a construção dos clusters, é razoável para os casos de matrizes com poucos atributos. Mas, nos casos de matrizes com muitos atributos, principalmente quando estes são heterogêneos, esse procedimento pode não ter um desempenho aceitável (TANAY *et al.*, 2002). Além do mais, esta técnica implica na associação de um objeto a apenas um cluster e, muitas vezes, isto não expressa a realidade, na qual um objeto pode fazer parte de nenhum, um ou mais de um agrupamento. É especialmente nestes aspectos que a biclusterização traz vantagens em relação à clusterização, pois esta pode agrupar os objetos com base em apenas um subconjunto de atributos, sendo que este subconjunto pode ser distinto para cada *bicluster* encontrado dentro da matriz de dados. Métodos de clusterização usuais não identificam adequadamente este tipo de correlação local (DE FRANÇA *et al.*, 2006; DE CASTRO *et al.*, 2007a). Além disso, cada objeto pode fazer parte de mais de um bicluster, e fazê-lo com base em um subconjunto distinto de atributos em cada bicluster.

O termo *biclusterização* foi introduzido por MIRKIN (1996) para descrever a clusterização simultânea dos conjuntos de linhas e colunas de uma matriz de dados. Mais recentemente, o termo foi utilizado por CHENG & CHURCH (2000) na análise de dados de expressão gênica, os quais foram os responsáveis pela popularização das técnicas de biclusterização com seu algoritmo denominado CC. Entretanto, HARTIGAN (1972) foi o primeiro a propor um algoritmo que realiza simultaneamente a clusterização de linhas e colunas de uma matriz de dados,

utilizando o termo *direct clustering*. Outros termos que podem ser encontrados na literatura são: *coclustering*, *two-way clustering*, *bidimensional clustering*, *subspace clustering*, dentre outros (MADEIRA & OLIVEIRA, 2004).

Como CHENG & CHURCH (2000) aplicaram a biclusterização em dados de expressão gênica, definiram o bicluster como sendo um subconjunto de genes e um subconjunto de condições, com um alto grau de *correlação*. Generalizando, pode-se dizer que um bicluster é um subconjunto de objetos e um subconjunto de atributos, com um alto grau de correlação. Esta correlação é uma medida de coerência entre os objetos e os atributos no bicluster. Um exemplo é a *coerência aditiva*, que será explicada na Seção 4.3.

A aplicação da biclusterização é muito ampla e não se reduz apenas à biologia, apesar de a maioria das aplicações encontradas na literatura serem em matrizes de dados de expressão gênica. Na realidade, esta matriz pode representar diferentes tipos de dados numéricos, como objetos e atributos (compreendendo as linhas e colunas da matriz) ou um mapa de cores de uma imagem (DE FRANÇA *et al.*, 2006). Como exemplo de outras áreas, pode-se citar: recuperação de informação e mineração de textos (DE CASTRO *et al.*, 2007c; FELDMAN & SANGER, 2007); filtragem colaborativa (DE CASTRO *et al.*, 2007b; DE CASTRO *et al.*, 2007d; SYMEONIDIS *et al.*, 2007); redução de dimensionalidade (AGRAWAL *et al.*, 1999); e análise de dados eleitorais (HARTIGAN, 1972). Na filtragem colaborativa, por exemplo, a biclusterização pode ser muito eficiente, podendo identificar subgrupos de clientes com preferências ou comportamentos similares em relação a um subconjunto de produtos. Essa informação pode ser utilizada em sistemas de recomendação e/ou definição de público-alvo. Em mineração de textos, o objetivo é identificar subgrupos de documentos com propriedades similares em relação a subgrupos de atributos, como palavras ou imagens. Este tipo de informação pode ser muito importante na consulta e indexação no domínio das máquinas de busca.

4.1 Formalização do Problema

Considere novamente a matriz A, contendo M linhas e N colunas. Cada elemento a_{ij} dessa matriz é um valor que representa a relação existente entre a linha i e a coluna j. Por exemplo, considere que cada linha da matriz representa um cliente de uma determinada vídeo-locadora e

cada coluna representa um filme existente nesta vídeo-locadora. Sendo assim, cada elemento a_{ij} desta matriz poderia ser a nota atribuída pelo cliente i ao assistir o filme j.

O conjunto de linhas da matriz A é dado por $L = \{1, 2, ..., M\}$ e o conjunto de colunas é dado por $C = \{1, 2, ..., N\}$. Será utilizado $A(L, C)$ para denotar a matriz A. Considerando que $I \subseteq L$ e $J \subseteq C$ são, respectivamente, subconjuntos de linhas e colunas, $A(I, J)$ denota a submatriz de A com o conjunto de linhas I e o conjunto de colunas J, onde $I = \{i_1, i_2, ..., i_k\}$ é um subconjunto de linhas ($I \subseteq L$ e $k \leq M$) e $J = \{j_1, j_2, ..., j_s\}$ é um subconjunto de colunas ($J \subseteq C$ e $s \leq N$).

Com base nesta notação, um bicluster pode ser interpretado como uma submatriz $A(I, J)$ com um subconjunto de linhas que apresentam comportamento semelhante em um subconjunto de colunas, e vice-versa. Portanto, dada uma matriz de dados, A, um algoritmo de biclusterização deve descobrir um conjunto de biclusters $\{B_1, B_2,..., B_P\}$, com $B_l = (I_l, J_l)$ e $l = 1, 2, ..., P$, de tal modo que cada bicluster B_l satisfaça algumas características específicas de homogeneidade (as quais variam de abordagem para abordagem) (MADEIRA & OLIVEIRA, 2004).

Existem diversas abordagens para a resolução deste problema. Uma delas estabelece uma conexão entre a resolução de um problema de biclusterização e um problema de grafo bipartido, devido à similaridade entre esses dois problemas. Para tal, considere $G = (V, E)$ um *grafo bipartido ponderado*, onde V é o conjunto de vértices e E é o conjunto de arestas. Um grafo é dito ser bipartido se os seus vértices podem ser particionados em dois conjuntos L e R, de tal forma que cada aresta existente em E possui exatamente uma ponta em L e a outra em R: $V = L \cup R$. A matriz de dados $A(L, C)$ pode ser vista como um grafo bipartido ponderado, onde cada nó $n_i \in L$ corresponde a uma linha e cada nó $n_j \in R$ corresponde a uma coluna. A aresta entre os nós n_i e n_j possui um peso a_{ij}, denotando o elemento da matriz na intersecção entre a linha i e a coluna j. Devido a esta similaridade, pode-se dizer que encontrar um bicluster de tamanho máximo é equivalente a encontrar um biclique (*grafo bipartido completo*, isto é, um caso especial de grafo bipartido, em que todos os nós $n_i \in L$ são conectados a todos os nós $n_j \in R$) máximo (CHENG & CHURCH, 2000; TANAY *et al.*, 2002; MADEIRA & OLIVEIRA, 2004). Enfim, a biclusterização pode ser interpretada como um problema de bipartição, o qual é claramente um problema de otimização combinatória.

Na literatura, encontram-se muitos algoritmos para a resolução do problema da biclusterização. Para avaliá-los, é necessário saber a necessidade do usuário e levar em consideração (MADEIRA & OLIVEIRA, 2004):

- O tipo de biclusters que o algoritmo pode encontrar. Os tipos de biclusters são estudados na Seção 4.2.
- A estrutura dos biclusters produzidos. Basicamente, um algoritmo de biclusterização pode produzir: um único bicluster (como na Figura 4.1.a); ou *P* biclusters (como nas Figuras 4.1.b a 4.1.i), sendo que *P* é o número de biclusters que se deseja procurar e, normalmente, é definido a priori.
- A estratégia utilizada pelo algoritmo para identificar cada bicluster, o que é estudado na Seção 4.3.
- O domínio das aplicações de cada algoritmo, como análise de dados biológicos, filtragem colaborativa, mineração de textos, etc.

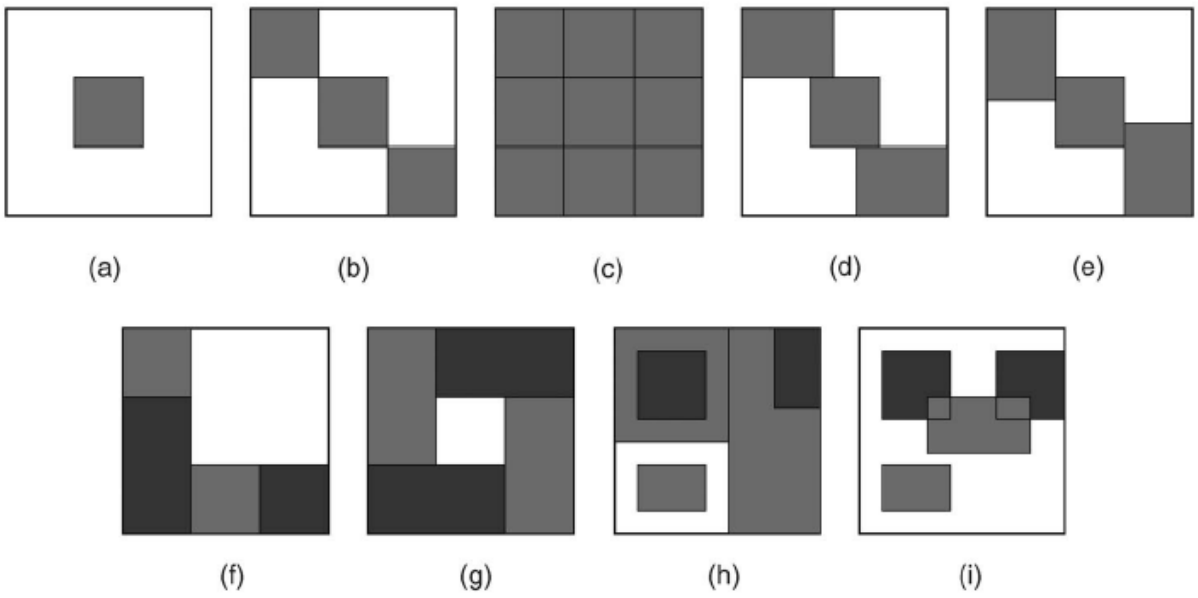


Figura 4.1: Estruturas dos biclusters. (a) bicluster único; (b) biclusters com linhas e colunas exclusivas; (c) estrutura de xadrez; (d) biclusters com linhas exclusivas; (e) biclusters com colunas exclusivas; (f) biclusters não-sobrepostos com estrutura de árvore; (g) biclusters não-sobrepostos não-exclusivos; (h) biclusters sobrepostos com estrutura hierárquica; e (i) biclusters sobrepostos arbitrariamente posicionados.
Fonte: MADEIRA & OLIVEIRA, 2004.

A Figura 4.1 pode induzir a crer que cada bicluster é contíguo na matriz de dados original, mas isso não é verdade. Um bicluster é formado por um subconjunto de objetos e um subconjunto de atributos, os quais apresentam similaridade alta entre si e não precisam ser contíguos na matriz original. A Figura 4.2 traz um exemplo de um bicluster que não é contíguo.

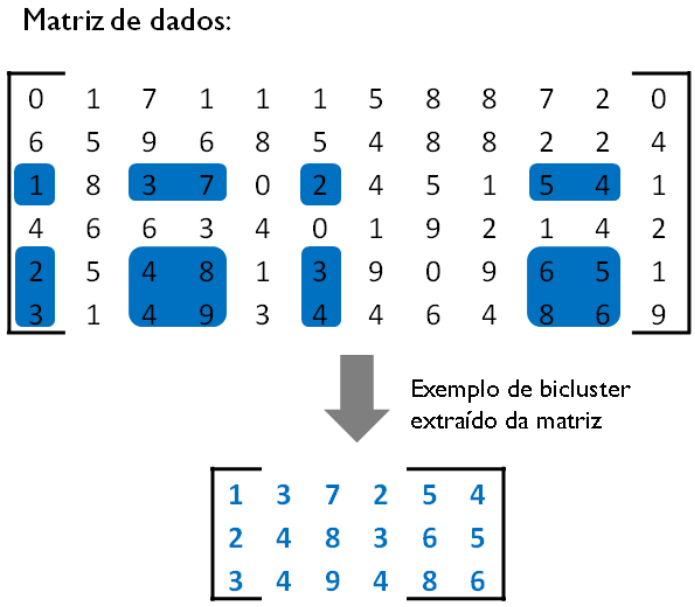


Figura 4.2: Exemplo de um bicluster formado por um subconjunto de objetos e um subconjunto de atributos não-contíguos na matriz de dados original.

Embora a complexidade do problema de biclusterização possa depender da formulação do problema e, especificamente, da função de avaliação utilizada para avaliar a qualidade de um dado bicluster, praticamente todas as variantes interessantes deste problema são NP-completas. (MADEIRA & OLIVEIRA, 2004).

4.2 Tipos de biclusters

Um critério interessante para avaliar um algoritmo de biclusterização é o tipo de bicluster que ele é capaz de produzir. MADEIRA & OLIVEIRA (2004) dividiram os biclusters em cinco tipos principais:

1. Biclusters com Valores Constantes (BVC);
2. Biclusters com Valores Constantes nas Linhas (BVCL);
3. Biclusters com Valores Constantes nas Colunas (BVCC);
4. Biclusters com Valores Coerentes (BVCo); e
5. Biclusters com Evoluções Coerentes (BEC).

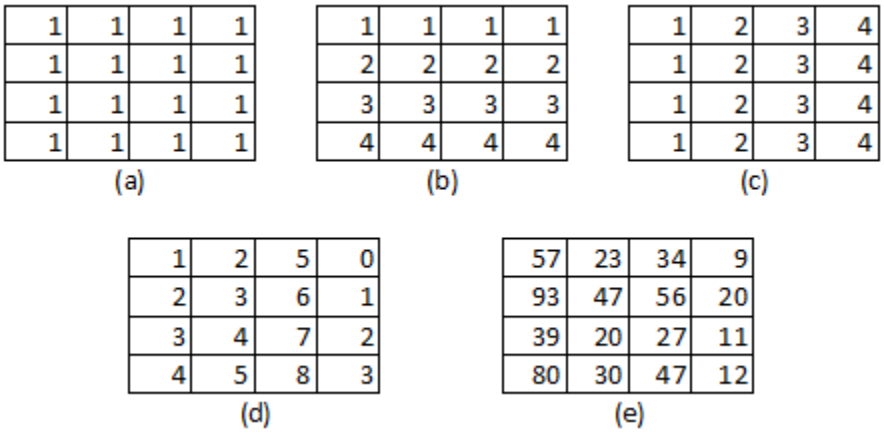


Figura 4.3: Exemplos de tipos diferentes de biclusters. (a) BVC. (b) BVCL. (c) BVCC. (d) BVCo. (e) BEC. Baseado em MADEIRA & OLIVEIRA (2004).

Os quatro primeiros tipos de biclusters, BVC, BVCL, BVCC e BVCo, ilustrados nas Figuras 4.3.a a 4.3.d, respectivamente, apresentam subconjuntos de linhas e colunas com comportamentos similares. O caso mais geral desses três é o BVCo, sendo que os outros são apenas casos especiais deste, como será visto na Seção 4.3. O quinto tipo, BEC, ilustrado na Figura 4.3.e, apresenta comportamento coerente, independentemente dos valores numéricos exatos na matriz de dados. Isso significa que, diferentemente dos quatro primeiros tipos, o BEC não busca valores coerentes dentro dos biclusters, mas sim propriedades coerentes nos valores (DE FRANÇA, 2010). Veja no exemplo da Figura 4.3.e que, apesar das colunas da matriz não apresentarem valores coerentes, elas apresentam o seguinte comportamento: $a_{i4} \leq a_{i2} \leq a_{i3} \leq a_{i1}$, com $i = 1, 2, 3$ ou 4.

Como o algoritmo de biclusterização utilizado neste trabalho objetiva encontrar BVCo, eles são explicados com mais detalhes na próxima seção.

4.3 Biclusters com Valores Coerentes

Um *Bicluster com Valores Coerentes* (BVCo) *perfeito* é uma submatriz $A(I, J)$ em que todos os seus valores podem ser obtidos em função de ajustes da linha e da coluna do elemento, conforme exemplificado a seguir:

$$a_{ij} = \mu + \alpha_i + \beta_j, \tag{4.1}$$

$$a_{ij} = \mu * \alpha_i * \beta_j, \tag{4.2}$$

onde μ é o valor-base do bicluster, α_i é o ajuste para a linha $i \in I$ e β_j é o ajuste para a coluna $j \in J$. Os ajustes exemplificados acima são obtidos pelos modelos aditivo (Equação 4.1) e multiplicativo (Equação 4.2), respectivamente. Como este trabalho foca na utilização do modelo aditivo, todo o restante do texto considera esse modelo.

Como já foi dito, os biclusters do tipo BVC, BVCL e BVCC são casos especiais de BVCo. Para BVC, $\alpha_i = 0$ e $\beta_j = 0$. Para BVCL, $\beta_j = 0$. Para BVCC, $\alpha_i = 0$.

Podem-se descobrir os valores de μ, α_i e β_j através do valor médio dos elementos do bicluster (a_{IJ}), do valor médio dos elementos na linha i (a_{iJ}) e do valor médio dos elementos na coluna j (a_{Ij}):

$$a_{iJ} = \mu + \alpha_i + \overline{\beta} = \frac{1}{|J|}\sum_{j \in J} a_{ij}, \tag{4.3}$$

$$a_{Ij} = \mu + \overline{\alpha} + \beta_j = \frac{1}{|I|}\sum_{i \in I} a_{ij}, \tag{4.4}$$

$$a_{IJ} = \mu + \overline{\alpha} + \overline{\beta} = \frac{1}{|I||J|}\sum_{i \in I, j \in J} a_{ij} = \frac{1}{|I|}\sum_{i \in I} a_{iJ} = \frac{1}{|J|}\sum_{j \in J} a_{Ij}, \tag{4.5}$$

onde $|\zeta|$ é a quantidade de elementos do conjunto ζ, $\overline{\alpha}$ é a média do termo de ajuste aditivo das linhas e $\overline{\beta}$ é a média do termo de ajuste aditivo das colunas.

Isolando os valores de μ, α_i e β_j nas Equações 4.3, 4.4 e 4.5, obtém-se:

$$\alpha_i = a_{iJ} - \mu - \overline{\beta}, \tag{4.6}$$

$$\beta_j = a_{Ij} - \mu - \overline{\alpha}, \tag{4.7}$$

$$\mu = a_{IJ} - \overline{\alpha} - \overline{\beta}. \tag{4.8}$$

Substituindo a Equação 4.8 nas Equações 4.6 e 4.7, obtém-se:

$$\alpha_i = a_{iJ} - a_{IJ} + \overline{\alpha},. \tag{4.9}$$

$$\beta_j = a_{Ij} - a_{IJ} + \overline{\beta}. \tag{4.10}$$

E, finalmente, substituindo as Equações 4.8, 4.9 e 4.10 na Equação 4.1, obtém-se:

$$a_{ij} = a_{Ij} + a_{iJ} - a_{IJ}. \tag{4.11}$$

Ao contrário da Equação 4.1, o valor de a_{ij} pode ser facilmente calculado através da Equação 4.11, pois podem-se obter os valores de a_{iJ}, a_{Ij}, a_{IJ} através das Equações 4.3, 4.4 e 4.5, respectivamente.

Para avaliar a qualidade dos biclusters, CHENG & CHURCH (2000) propuseram o *resíduo quadrático médio* (MSR – do inglês *mean squared residue*):

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r_{ij}^2, \tag{4.12}$$

onde:

$$r_{ij} = a_{ij} - a_{Ij} - a_{iJ} + a_{IJ}. \tag{4.13}$$

De acordo com este critério, quanto menor for o valor do MSR, melhor a qualidade do bicluster. Logo, um bicluster perfeito é uma submatriz com MSR igual a zero. Entretanto, em matrizes de dados reais, dificilmente serão encontrados biclusters perfeitos. Normalmente, eles são mascarados por ruído. Por isso, os algoritmos de biclusterização, normalmente, não procuram biclusters perfeitos, e sim com alta similaridade, os quais CHENG & CHURCH (2000) chamaram de *δ-biclusters*. Uma submatriz $A(I, J)$ é considerada um δ-bicluster se $H(I, J) < \delta$ para algum $\delta \geq 0$. Desse modo, para todos os P biclusters que se deseja encontrar, um algoritmo de biclusterização objetiva minimizar o valor de $H(I, J)$ até um valor aceitável.

Porém, a fim de ser mais funcional e permitir uma análise mais elaborada dos dados, também é exigido que um bicluster apresente um grande volume (DE CASTRO *et al.*, 2007a). O volume de um bicluster é normalmente definido pela quantidade de células que o compõe, ou seja, $|I| \times |J|$. Logo, outro importante objetivo de um algoritmo de biclusterização é maximizar este volume para todos os P biclusters que se deseja encontrar.

Segundo DE CASTRO *et al*. (2007a), além destas características individuais dos biclusters (resíduo e volume), que devem ser otimizadas durante a sua construção, vários outros aspectos devem ser considerados:

- *Cobertura do conjunto de dados*: a quantidade de elementos da matriz de dados que são cobertos pelos biclusters gerados, ou seja, que tomam parte em pelo menos um bicluster.
- *Grau de sobreposição entre biclusters*: sobreposição entre os elementos dos biclusters obtidos. O grau de sobreposição de um bilcuster B_l dado um bicluster B_k é:

$$Sobreposição(k,l) = \frac{\left|I_k \cap I_l\right| \cdot \left|J_k \cap J_l\right|}{\left|I_l\right| \cdot \left|J_l\right|}. \tag{4.14}$$

  O grau de sobreposição entre todos os biclusters obtidos é dado pela média da sobreposição par a par.
- *Grau de ocupação do bicluster*: em matrizes com alta esparsidade, a quantidade de valores não-nulos presentes no bicluster.

Logo, percebe-se que a biclusterização é um problema de múltiplos objetivos e, evidentemente, os objetivos principais – de minimizar resíduo e maximizar volume – são conflitantes (quanto maior o volume, menor a probabilidade de haver alta coerência entre os elementos do bicluster), fato comum à maioria dos problemas de otimização multiobjetivo. É importante lembrar que, para um problema de otimização multiobjetivo, não existe uma única solução ótima, e sim *soluções não-dominadas* que se encontram na *fronteira de Pareto* (DEB *et al.*, 2002).

Apesar de a biclusterização ser um problema multiobjetivo, poucos artigos da literatura adotam conceitos de uma abordagem multiobjetivo (MITRA & BANKA, 2006). Um problema em comum com essas propostas é que, para a biclusterização, além das soluções não-dominadas, também é recomendado retornar biclusters relativos a outras áreas da matriz de dados original, procurando exibir as diferentes correlações do conjunto de dados (COELHO *et al.*, 2009).

Outro critério importante para a avaliação de um algoritmo de biclusterização é a técnica que ele utiliza para encontrar os biclusters, essas técnicas serão explicadas na Seção 4.4.

4.4 Técnicas Utilizadas na Biclusterização

Além dos tipos de biclusters que um algoritmo de biclusterização pode produzir, outro critério interessante para avaliá-lo é a técnica que ele utiliza para encontrar os biclusters. As possíveis técnicas foram divididas em cinco tipos por MADEIRA & OLIVEIRA (2004):

1. *Clusterização Iterativa Combinada de Linhas e Colunas*: são aplicados algoritmos de clusterização nas linhas e colunas da matriz de dados, separadamente, e então os resultados são combinados usando algum tipo de procedimento iterativo.
2. *Dividir e Conquistar*: o problema é dividido em vários subproblemas que são similares ao original, porém menores. Estes subproblemas são resolvidos recursivamente e, então, as soluções são combinadas para se criar uma única solução para o problema original. Estes algoritmos têm a vantagem de serem muito rápidos. Porém, podem perder bons biclusters devido às divisões do problema original.
3. *Procura Iterativa Gulosa*: estes algoritmos são baseados na ideia de criar biclusters através da adição ou remoção de linhas / colunas (nós) dos biclusters, usando um critério que maximiza o ganho local, na esperança de que esta escolha levará a uma boa solução global. Logicamente, nem sempre isso acontece. Estes algoritmos também são rápidos.
4. *Enumeração Exaustiva dos Biclusters*: estes algoritmos são baseados na ideia de que os melhores biclusters só podem ser encontrados enumerando exaustivamente todos os possíveis biclusters existentes na matriz de dados. Desse modo, eles sempre encontram os melhores biclusters. Porém, devido à grande quantidade de biclusters possíveis, estes algoritmos precisam adotar algumas restrições junto ao tamanho dos biclusters e também da matriz original a ser analisada.
5. *Identificação dos Parâmetros da Distribuição*: estes algoritmos consideram que os biclusters são gerados usando um dado modelo estatístico e tentam identificar os parâmetros da distribuição que ajusta, da melhor maneira, os dados disponíveis, minimizando iterativamente certo critério.

Os algoritmos também diferem entre si pela quantidade de biclusters que eles conseguem encontrar por execução. Alguns algoritmos encontram apenas um bicluster; outros encontram vários biclusters simultaneamente; e outros encontram pequenos conjuntos de biclusters (MADEIRA & OLIVEIRA, 2004; DE FRANÇA *et al.*, 2006).

Além disso, existem algoritmos determinísticos e não-determinísticos. Os algoritmos não-determinísticos podem encontrar diferentes soluções a cada execução, enquanto os determinísticos sempre encontram a mesma solução (DE FRANÇA *et al.*, 2006).

4.5 Síntese do Capítulo

Este capítulo apresentou a técnica de biclusterização, a qual é capaz de extrair subconjuntos de linhas e colunas de uma matriz de dados, de modo que os elementos desses subconjuntos sejam, de alguma forma, similares. Para medir essa similaridade, foi apresentada a métrica chamada *coerência aditiva*, a qual foi proposta por CHENG & CHURCH (2000). Esses autores também propuseram medir a qualidade de um bicluster através do *resíduo quadrático médio* (MSR). Também foi mostrado que, para permitir uma análise mais elaborada dos dados, é necessário que o bicluster apresente um bom volume.

Desse modo, um bom algoritmo de biclusterização produz biclusters com baixo MSR e grande volume. Porém, existem outras características que podem ser levadas em consideração no momento de avaliar um algoritmo de biclusterização, como o tipo de bicluster que ele produz e a cobertura do conjunto de dados.

No próximo capítulo, o problema de dados faltantes é abordado novamente apresentando três algoritmos de imputação: o KNNImpute, o rSVD e o SwarmBcluster. O SwarmBcluster é um algoritmo de biclusterização com a capacidade de obter um conjunto de biclusters que maximiza a cobertura dos dados e que, posteriormente, foi especializado para o tratamento de dados faltantes (DE FRANÇA, 2010). Ele é capaz de realizar a biclusterização em uma base de dados incompleta e utilizar as informações contidas em um bicluster para estimar seus valores ausentes. Para realizar a biclusterização em uma base de dados incompleta, o SwarmBcluster primeiramente realiza uma pré-imputação. Essa pré-imputação consiste em utilizar uma técnica simples e computacionalmente barata para realizar as imputações dos dados faltantes, como a imputação através da mediana ou do vizinho mais próximo. Utilizando a capacidade da biclusterização em gerar modelos com ruído reduzido (DE FRANÇA, 2010), a base de dados pré-imputada é analisada, de modo que sejam gerados biclusters que cubram o máximo possível dos dados faltantes originais. De posse dos biclusters, os valores pré-imputados são descartados e os dados faltantes são reestimados, de modo que cada um dos biclusters encontrados se torne o mais coerente possível.

Capítulo 5

Algoritmos utilizados nos experimentos de imputação única

Este capítulo descreve os três algoritmos utilizados nos testes de imputação única de dados faltantes realizados neste trabalho: KNNImpute, rSVD e SwarmBcluster.

O KNNImpute foi escolhido, porque os algoritmos baseados em *K*-NN são amplamente difundidos (FARHANGFAR *et al.*, 2004; FARHANGFAR *et al.*, 2007; ACUNA & RODRIGUEZ, 2004; HASTIE *et al.*, 1999; TROYANSKAYA *et al.*, 2001; COLANTONIO *et al.*, 2010; PATEREK, 2007; ZHANG *et al.*, 2006; BROCK *et al.*, 2008; MOHAMMADI & SARAEE, 2008; ZHAO *et al.*, 2007) e tendem a gerar bons resultados (TROYANSKAYA *et al.*, 2001; COLANTONIO *et al.*, 2010; ZHANG *et al.*, 2006; BROCK *et al.*, 2008; MOHAMMADI & SARAEE, 2008; ZHAO *et al.*, 2007).

O algoritmo rSVD foi escolhido pelos mesmos motivos de KNNImpute, ou seja, os algoritmos baseados em SVD também são amplamente difundidos (HASTIE *et al.*, 1999; TROYANSKAYA *et al.*, 2001; PATEREK, 2007; FUNK, 2006; BROCK *et al.*, 2008; ZHAO *et al.*, 2007) e tendem a gerar bons resultados (TROYANSKAYA *et al.*, 2001; PATEREK, 2007; FUNK, 2006; BROCK *et al.*, 2008; ZHAO *et al.*, 2007; KOREN & BELL, 2011). Além disso, o rSVD obteve grande destaque em uma competição internacional de sistemas de recomendação, promovida pela empresa Netflix.

O terceiro algoritmo, SwarmBcluster, é o único dos três a utilizar a biclusterização para a imputação. Através dele, pretende-se demonstrar a eficácia deste paradigma para a imputação de dados faltantes.

5.1 O algoritmo KNNImpute

Os métodos baseados em *K-vizinhos mais próximos* (*K-NN* – do inglês *K-Nearest Neighbors*) selecionam *K* objetos com perfis semelhantes ao objeto de interesse para substituir os valores faltantes. Portanto, supondo que um objeto *o* possui um valor faltante no atributo *a*, o

método KNNImpute encontrará os *K* objetos mais similares (levando em consideração os demais atributos da matriz de dados) a este objeto *o*. Logicamente, estes *K* objetos não devem possuir dado faltante para o atributo *a*. Desta maneira, independentemente da métrica utilizada para se medir a similaridade, o dado faltante pode ser estimado através da média ponderada dos valores atribuídos pelos *K* objetos ao atributo *a*:

$$p_{oa} = \frac{\sum_{\{u \in T_o | a \in S_u\}} sim(o,u) * r_{ua}}{\sum_{\{u \in T_o | a \in S_u\}} sim(o,u)}, \tag{5.1}$$

onde:

- T_o é o conjunto de *K* vizinhos do objeto *o.*
- S_u é o conjunto de atributos que não possuem valor faltante para o objeto *u*.
- *sim*(*o,u*) é a similaridade (calculada através da métrica escolhida) existente entre o objeto *o* e o objeto *u*.
- r_{ua} é o valor atribuído pelo objeto *u* ao atributo *a*.

Uma forma de retirar parte do viés dessa estimativa pode ser realizada utilizando os valores médios do objeto e do atributo em questão, conforme a Equação 5.2:

$$p_{oa} = \bar{r}_o + \frac{\sum_{\{u \in T_o | a \in S_u\}} sim(o,u) * (r_{ua} - \bar{r}_u)}{\sum_{\{u \in T_o | a \in S_u\}} sim(o,u)}, \tag{5.2}$$

onde: $\bar{r}_o$ e $\bar{r}_u$ são as médias dos valores atribuídos aos atributos dos objetos *o* e *u*, respectivamente. Logo:

$$\bar{r}_o = \frac{\sum_{\{i \in S_o\}} r_{oi}}{|S_o|}, \bar{r}_u = \frac{\sum_{\{i \in S_u\}} r_{ui}}{|S_u|}, \tag{5.3}$$

onde $|\zeta|$ é a quantidade de elementos do conjunto ζ.

Suponha que a base de dados, junto à qual os valores serão estimados, representa a opinião de clientes (objetos) sobre filmes (atributos) assistidos, e que esta opinião é dada por uma nota que varia de 1 a 5. Um usuário pode dar nota 4 a um filme que gosta e 1 para um filme que não gosta. Outro usuário pode dar nota 5 a um filme que gosta e 2 para um filme que não gosta. Neste cenário e em cenários semelhantes, CANDILLIER *et al.* (2008) afirmam que a Equação 5.2 gera resultados melhores que a Equação 5.1. Neste trabalho, a Equação 5.2 foi utilizada para a implementação do KNNImpute. Note, porém, que essa solução não é adequada em situações onde o domínio de valores dos atributos é diferente.

Existem várias métricas para determinar a similaridade entre dois objetos, como distância euclidiana, correlação de Pearson, índice de Jaccard, Chebychev, Hamming e Minkowski. Algumas medidas de similaridade, como a correlação de Pearson, calculam a similaridade entre dois objetos considerando apenas os atributos não-faltantes em comum entre esses dois objetos. Desse modo, dois vetores podem ser completamente similares, mesmo que eles compartilhem o mesmo valor para um único atributo. Outras medidas, como a Jaccard, medem a sobreposição que dois vetores compartilham em relação a seus atributos não-faltantes e não levam em conta os valores dos atributos. Desse modo, dois vetores podem ser completamente similares apenas se não possuírem nenhum dado faltante. Uma forma de superar essas dificuldades é combinando estes dois tipos de medidas de similaridade. Por isso, neste trabalho optou-se por utilizar a correlação de Pearson (STIGLER, 1989) combinada com o índice de Jaccard (JACCARD, 1901) e, assim, aproveitar a complementaridade destas duas medidas. A medida Jaccard servirá como um peso para Pearson. Logo, o resultado final da similaridade será dado pelo produto de Jaccard e Pearson. As Equações 5.4 e 5.5 são, respectivamente, as fórmulas para Pearson e Jaccard:

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{\left(\sum X\right)^2}{N}\right)\left(\sum Y^2 - \frac{\left(\sum Y\right)^2}{N}\right)}}, \tag{5.4}$$

$$S = \frac{|X \cap Y|}{|X \cup Y|}, \tag{5.5}$$

onde:

- X, Y são dois vetores numéricos de mesma dimensão, que representam dois objetos da base de dados*;*
- N é a quantidade de atributos não-faltantes dos objetos X e Y; e
- $|\zeta|$ é a quantidade de elementos do conjunto ζ.

Os resultados da correlação de Pearson variam entre −1 e +1, sendo que +1 representa uma correlação positiva perfeita, −1 uma correlação negativa perfeita e 0 implica independência total entre os dois objetos analisados. De forma a manter o valor consistente, o cálculo desse

coeficiente é efetuado tomando apenas os elementos em que os valores não são faltantes, em ambos os vetores.

Com relação ao índice de Jaccard, quanto maior o seu resultado, mais similares são os dois objetos. Para a situação de dados faltantes, o índice de Jaccard medirá a interseção e união do conjunto de elementos não-faltantes dentro dos dois objetos.

Assim, o algoritmo KNNImpute pode ser descrito da seguinte maneira:

```
Algoritmo 5.1: [IMP] = KNNImpute(A, K)
% A – matriz com dados faltantes
% K – quantidade desejada de vizinhos
% IMP – matriz com os dados imputados
IMP ← A
vs ← calcula_similaridade(A)
Para cada dado faltante (i,j) de IMP faça
    v ← KNN(A, vs, (i,j), K)
    (i,j)← poa(v)
Fim Para
```

onde:

- *calcula_similaridade* é uma função que calcula a similaridade entre todas as linhas da matriz *A*;
- *KNN* é uma função que encontra os *K* vizinhos mais próximos para a imputação do dado faltante (*i, j*); e
- *poa* é a função que calcula o valor a ser imputado, o qual pode ser obtido através da Equação 5.1 ou 5.2. Lembrando que, neste trabalho, a Equação 5.2 foi utilizada.

Um dos pontos críticos do algoritmo KNNImpute é a escolha do valor do parâmetro *K*. Por um lado, se *K* tiver um valor muito grande, linhas que são significativamente diferentes podem ser levadas em consideração, podendo diminuir a acurácia da imputação (COLANTONIO *et al.*, 2010). Por outro lado, se *K* tiver um valor muito pequeno, isto é, se não for considerado um número suficiente de linhas relevantes, uma ênfase excessiva pode ser dada a padrões bem particulares encontrados na base de dados (COLANTONIO *et al.*, 2010). Note que a seleção do melhor *K* corresponde a operações adicionais, aumentando o custo computacional. Por isso, é mais viável conhecer a base de dados para se fazer uma boa inferência do valor de *K*. Para encontrar um valor de *K* apropriado para os experimentos deste trabalho, foram realizados alguns experimentos preliminares com as bases de dados utilizadas. Esses testes foram realizados para

algumas quantidades de dados faltantes e o melhor valor encontrado de *K* foi escolhido para realizar os demais testes.

Outra questão levantada por CANDILLIER *et al.* (2008) é que a predição pode ser feita na matriz transposta. Isso significa que o cálculo da similaridade será feito com base nos atributos. Essa forma trouxe melhores resultados nos experimentos realizados por CANDILLIER *et al.* (2008) e, como em experimentos preliminares nas bases de dados aqui utilizadas também ocorreu uma melhora no desempenho, esta será a forma utilizada neste trabalho.

5.2 O algoritmo rSVD

FUNK (2006) foi o primeiro a utilizar o algoritmo de *Decomposição em Valores Singulares Regulado* (rSVD – do inglês *regularized Singular Value Decomposition*) em filtragem colaborativa (PATEREK, 2007). Ele participava de uma competição chamada *Netflix Prize*, com maiores informações disponíveis em: http://netflixprize.com/ (BENNETT & LANNING, 2007). Os organizadores dessa competição disponibilizaram uma base de dados com mais de 100 milhões de avaliações de filmes feitas por 480.189 usuários. O objetivo era melhorar substancialmente a precisão das previsões sobre o quanto um usuário poderia gostar de um filme que ainda não assistiu. No início do torneio, o rSVD conquistou a posição de 3º lugar e, subsequentemente, se tornou o principal componente de outras abordagens utilizadas por outros competidores, inclusive dos dois algoritmos vencedores do torneio (DE FRANÇA, 2010).

A proposta de FUNK (2006) se baseou no fato de que a avaliação do usuário *i* para o filme *j* pode ser dada através da soma das preferências sobre diversos aspectos (características) do filme (Equação 5.6). Esses aspectos são considerados para todos os filmes e não necessariamente correspondem a atributos objetivos dos filmes. Os *F* aspectos podem assim ser interpretados como atributos abstratos que caracterizam um filme. Logo, cada filme deve ser descrito por *F* valores, representando cada um de seus aspectos, e cada usuário também é descrito por *F* valores, representando o quanto aprecia cada aspecto.

$$\hat{y}_{ij} = \sum_{k=1}^{F} u_{ki} v_{kj} \tag{5.6}$$

onde:

- u_{ki} é o quanto o usuário *i* aprecia o aspecto *k*; e

- v_{kj} é o valor atribuído ao aspecto k do filme j.

Como é possível perceber, por meio desta técnica, a matriz original, na qual as linhas representam os usuários e as colunas representam os filmes, é decomposta em duas matrizes menores. A técnica de SVD é utilizada para encontrar estas duas matrizes menores. Para isso, será tomado o *rank-F* SVD da matriz original, obtendo, assim, a melhor aproximação (menor erro), dentro dos limites desse modelo (FUNK, 2006).

Apesar de FUNK (2006) ter desenvolvido este algoritmo para a competição *Netflix Prize*, ele é um algoritmo genérico para o problema de imputação ou estimação de dados faltantes. Basicamente, ele pode ser utilizado para qualquer base de dados em que as linhas representam objetos e as colunas representam atributos.

Dois problemas para a utilização da técnica de SVD na competição *Netflix Prize* foram: (*i*) a dimensão da base de dados e (*ii*) a quantidade de dados faltantes da base de dados. Para superar esses desafios, FUNK (2006) calculou a SVD utilizando o método do gradiente descendente, ou seja, seguindo a derivada do erro de aproximação, que tem a vantagem adicional de tornar possível simplesmente ignorar os erros desconhecidos devido aos dados faltantes.

O erro nas predições é dado simplesmente por:

$$E = y_{ij} - \hat{y}_{ij}, \tag{5.7}$$

onde y_{ij} é o valor real do atributo j do objeto i e $\hat{y}_{ij}$ é o valor predito do atributo j do objeto i.

O gradiente descendente calcula a derivada parcial do erro quadrático com relação a cada parâmetro, isto é, com relação a cada u_{ki} e v_{kj}. Logo, para calcular isso para um dado particular do objeto $i = I$ para o atributo $j = J$, considerando apenas um aspecto singular $k = K$, com relação a u_{KI}, tem-se (FUNK, 2007):

$$\frac{\partial E^2}{\partial u_{KI}} = 2 * E * \frac{\partial E}{\partial u_{KI}} = 2 * E * (-v_{KJ}) \tag{5.8}$$

A Equação 5.8 calcula a derivada para uma avaliação particular. Logo, a derivada completa é dada pelo somatório dos resultados da Equação 5.8 sobre todas as avaliações realizadas pelo usuário I. A mesma ideia deve ser utilizada para calcular a derivada parcial com relação a v_{KJ}, logo:

$$\frac{\partial E^2}{\partial v_{KJ}} = 2 * E * \frac{\partial E}{\partial v_{KJ}} = 2 * E * (-u_{KI}) \tag{5.9}$$

Como FUNK (2006) utiliza a regra delta de aprendizado para o gradiente descendente, para a atualização de u_{KI} e v_{KJ}, é utilizado um parâmetro arbitrário, chamado de *taxa de aprendizagem* (*lrate*):

$$u_{KI} = u_{KI} - lrate * \frac{\partial E^2}{\partial u_{KI}} = u_{KI} + lrate * 2 * E * v_{KJ}, \quad (5.10)$$

$$v_{KJ} = v_{KJ} - lrate * \frac{\partial E^2}{\partial v_{KJ}} = v_{KJ} + lrate * 2 * E * u_{KI}. \quad (5.11)$$

Entretanto, além da técnica do gradiente descendente, FUNK (2006) também utilizou uma técnica de regularização, com o intuito de prevenir o *sobre-ajuste* ao aprender o modelo de correlações que melhor representa os valores conhecidos da base. Portanto, as equações realmente utilizadas para a atualização de u_{KI} e v_{KJ} são dadas por:

$$u_{KI} = u_{KI} + lrate * (2 * E * v_{KJ} - \lambda * u_{KI}), \quad (5.12)$$

$$v_{KJ} = v_{KJ} + lrate * (2 * E * u_{KI} - \lambda * v_{KJ}), \quad (5.13)$$

onde λ é a taxa de regularização. O Algoritmo 5.2 descreve o algoritmo rSVD.

Algoritmo 5.2: [*U*, *V*] = rSVD(*A*, *F*, *lrate*, *init_val*, λ, *max_it*, *min_improvement*)

```
% A - matriz com dados faltantes
% init_val - valor inicial das células das matrizes U e V
% Critérios de parada:
% max_it - número máximo de iterações
% min_improvement - valor mínimo de melhora significativa entre iterações
Inicialize U, V
Para cada um dos F aspectos faça
    Enquanto não atingir o critério de parada faça
        Para cada dado não-faltante (i,j) de A faça
            Calcule ŷ_ij (Equação 5.6)
            Calcule E (Equação 5.7)
            Atualize U_KI e V_KJ (Equações 5.12 e 5.13)
        Fim Para
    Fim Enquanto
Fim Para
```

O algoritmo rSVD retorna uma matriz *U* com *F* linhas e *N* colunas e uma matriz *V* com *F* linhas e *M* colunas. Uma estimativa da matriz de dados *A* pode ser obtida por $U^T \times V$. Porém, em vez de fazer $U^T \times V$, é menos custoso utilizar a Equação 5.6 para cada dado faltante que se deseja estimar.

Uma questão que ainda pode restar sobre o algoritmo rSVD é o porquê do retorno de apenas duas matrizes (U e V), visto que a técnica de SVD consiste em: dada uma matriz $A \in \Re^{Nxp}$ com posto $r \leq \min(N, p)$, então A pode ser expressa na forma $A = UDV^T$ com $U \in \Re^{Nxr}$, $V \in \Re^{pxr}$, $U^TU = V^TV = I_r$ e $D \in \Re^{pxr}$ uma matriz diagonal cujos elementos na diagonal são os valores singulares e obedecem $d_1 \geq d_2 \geq ... \geq d_r \geq 0$. Conforme relatado por FUNK (2006), os valores da matriz D estão inseridos nas matrizes U e V, mas podem ser facilmente extraídos, se necessário.

5.3 O algoritmo SwarmBcluster

O *SwarmBcluster* é um algoritmo de biclusterização baseado em *Inteligência de Enxames* (KENNEDY *et al.*, 2001) e proposto por DE FRANÇA & VON ZUBEN (2010). Esse algoritmo é capaz de encontrar um conjunto de biclusters que maximiza a cobertura da base de dados sem degradar de modo significativo o volume médio e o grau de coerência (medida associada ao MSR). Esta é uma característica importante, pois a maioria dos algoritmos de biclusterização objetiva apenas a minimização do MSR e a maximização do volume, gerando biclusters sobrepostos e que não cobrem amplamente a base de dados, o que pode gerar informações redundantes em algumas porções da base de dados e falta de informações em outras porções.

Por se tratar de um algoritmo baseado em Inteligência de Enxames, a proposta trabalha com um conjunto de agentes independentes e auto-organizáveis, que tendem a exibir um comportamento coerente através da interação entre si e com o ambiente, direta ou indiretamente. Com isso, esses agentes são capazes de conciliar dois objetivos de um problema de busca: explorar e explotar. A exploração é o 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. Já a explotação é o ato de buscar soluções dando um peso maior às informações adquiridas pelos agentes (DE FRANÇA, 2010). Dessa forma, esses agentes podem utilizar as informações globais para melhorar suas próprias soluções.

Especificamente, o SwarmBcluster é baseado no algoritmo de *Otimização por Colônias de Formigas* (ACO – do inglês *Ant Colony Optimization*) proposto por DORIGO (1992). Esse algoritmo é uma meta-heurística de *Inteligência de Enxames* baseada no estudo feito por GOSS *et al.* (1989) sobre o comportamento das formigas argentinas quando estão procurando comida. As formigas argentinas, assim como outras espécies de formigas, procuram a fonte de alimento

percorrendo trajetórias predominantemente aleatórias pelo espaço ao redor do ninho. Quando uma formiga encontra uma fonte de alimento, ela volta ao ninho depositando uma substância química denominada feromônio, deixando uma trilha de feromônio da fonte de alimento descoberta até o ninho. Sempre que uma formiga começa a procurar comida, ela tende a seguir uma trilha com probabilidade proporcional à intensidade de feromônio existente na trilha. Obviamente, as trilhas menores são privilegiadas, pois elas são percorridas mais rapidamente. Logo, num mesmo intervalo de tempo, o feromônio dessa trilha será reforçado mais vezes e, consequentemente, o efeito da evaporação do feromônio será atenuado. Portanto, como o caminho mais curto possui maior concentração de feromônio, é mais provável que as formigas o encontrem e o sigam.

Um problema clássico de menor caminho é o *Problema do Caixeiro Viajante* (TSP – do inglês *Traveling Salesman Problem*) (KRUSKAL JR, 1956). O objetivo deste problema é: dado um grafo ponderado $G = (V; E; H)$, composto por $| V |$ vértices, $| E | = | V | (| V | - 1) / 2$ arestas e uma matriz de pesos H, encontrar o circuito de menor comprimento que passa uma e somente uma vez em cada vértice. No ACO para TSP, a matriz de pesos, ou também chamada matriz de informação local, pode ser formada, por exemplo, pelo inverso da distância euclidiana entre os pares de vértices do grafo. A Figura 5.1 traz um exemplo de uma possível solução para um TSP: os pontos em azul são os vértices e em vermelho estão as arestas que formam o circuito encontrado.

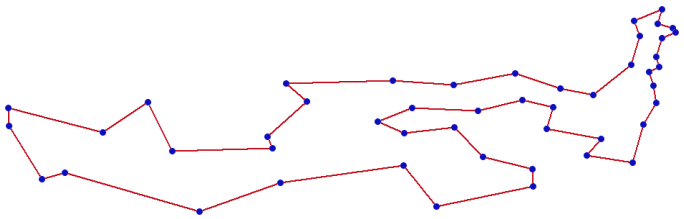


Figura 5.1: Exemplo de uma solução do TSP.

Para solucionar o TSP, cada formiga (agente) escolhe aleatoriamente um vértice para iniciar seu circuito e vai fazendo escolhas até passar por todos os vértices. Para fazer estas escolhas, além da informação local associada a cada aresta (matriz H), DORIGO (1992) também utilizou uma matriz de feromônio artificial, T, associando uma variável τ_{ij} a cada aresta. A probabilidade da formiga k que está no vértice i ir para o vértice j na iteração t é proporcional ao nível de feromônio e à informação local da aresta (i,j):

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{l \in J^k} \tau_{il}^{\alpha} * \eta_{il}^{\beta}}, \text{se } j \in J^k \\ 0, \quad \text{caso contrário} \end{cases}, \tag{5.14}$$

onde:

- τ_{ij} é o nível de feromônio da aresta (i,j);
- η_{ij} é a informação local da aresta (i, j), ou seja, o inverso da distância entre o vértice i e j;
- α e β são parâmetros definidos pelo usuário; e
- J^k é a lista de cidades que ainda podem ser visitadas pela formiga k.

Após todas as formigas terem construído seu percurso, a matriz de feromônio, T, é atualizada com base nas soluções geradas:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}, \tag{5.15}$$

onde ρ é a taxa de evaporação de feromônio e $\Delta\tau_{ij}$ é definido por:

$$\Delta\tau_{ij} = \begin{cases} \sum_{k \in K} \dfrac{1}{f(S_k)}, \text{se } (i,j) \in S_k \\ 0, \text{caso contrário} \end{cases}, \tag{5.16}$$

onde $f(S_k)$ é o valor da função-objetivo do problema para a solução completa S_k gerada pela formiga k. Note que essa fórmula generaliza outras formas de se calcular esse incremento. Por exemplo, em algumas variantes do ACO, K é o conjunto formado por uma porcentagem das melhores formigas; em outras variantes, K é um conjunto unitário com a melhor solução encontrada até o momento.

O Algoritmo 5.3 representa o algoritmo ACO. O critério de parada mais comumente utilizado é um número máximo de iterações (*max_it*).

Algoritmo 5.3: [melhor_solução] = ACO(α, β, ρ, n_ants)

```
% n_ants - quantidade de formigas utilizadas
Inicialize a matriz de feromônio T com probabilidades iguais
Enquanto o critério de parada não for atingido faça
    Para c = 1 até n_ants faça
        Construir_solução()
    Fim Para
    Atualizar_feromônio()
Fim Enquanto
```

A fim de se obter melhoria na qualidade das soluções, diversas modificações e extensões foram feitas na estrutura original do ACO. Três delas foram utilizadas para a criação do SwarmBcluster: o algoritmo *Max-Min Ant System* (MMAS), o algoritmo *Hyper-Cube Framework* para o ACO (HCF-ACO) e o algoritmo MMAS melhorado.

O algoritmo MMAS foi criado por STÜTZLE & HOOS (2000) com o intuito de evitar a convergência prematura do ACO original. A modificação feita nesse algoritmo é referente à atualização dos níveis de feromônio. No MMAS, o conjunto K da Equação 5.16 é composto apenas pela melhor solução global (melhor solução encontrada até o momento entre todas as iterações) e pela melhor solução local (melhor solução encontrada na iteração atual). A outra alteração, que deu origem ao nome desse método, impõe limites máximo (τ_{max}) e mínimo (τ_{min}) aos valores do feromônio. Se os valores de τ_{max} e τ_{min} forem desrespeitados ao se aplicar a Equação 5.15, o resultado é substituído por τ_{max} ou τ_{min}, dependendo de qual deles foi violado.

Apesar de efetivamente melhorar a robustez e flexibilidade, o MMAS ainda apresentava dificuldades quando aplicado a problemas com dimensão elevada. Por isso, BLUM & DORIGO (2004) propuseram o HCF-ACO. Essa proposta fixou os limites mínimo e máximo do feromônio no intervalo (0, 1). Para tanto a Equação 5.16 utiliza o valor normalizado referente à qualidade da solução, da seguinte maneira:

$$\Delta\tau_{ij} = \begin{cases} \sum_{k \in K} \dfrac{F(S_k)}{\sum_{k' \in K} F(S_{k'})}, & \text{se } (i,j) \in S_k \\ 0, & \text{caso contrário} \end{cases}, \qquad (5.17)$$

onde $F(S_k)$ é uma função que mede a qualidade objetivando a maximização. Além disso, a matriz de feromônio teve seus valores inicializados em 0,5. Segundo DE FRANÇA (2010), 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 explotar arestas da melhor solução, a probabilidade de desviar do caminho ótimo é alta o suficiente para evitar a convergência prematura.

Ainda buscando melhorias nos resultados, DE FRANÇA *et al*. (2004a, 2004b, 2005) propuseram o *MMAS melhorado*. A primeira modificação foi na Equação 5.17 do HCF-ACO, buscando distribuir de uma forma mais uniforme o incremento de feromônio pelas soluções geradas:

$$\Delta\tau_{ij} = \begin{cases} \sum_{k \in K} \dfrac{F(S_k) - F(S_{pior})}{F(S_{melhor}) - F(S_{pior})}, \text{se } (i,j) \in S_k \\ 0, \qquad \text{caso contrário} \end{cases}, \tag{5.18}$$

onde S_{melhor} e S_{pior} são a melhor e a pior soluções encontradas até o momento, respectivamente. Outra modificação realizada para evitar a convergência prematura foi a sua detecção através das seguintes equações:

$$\sum_{\tau_{ij} \geq \tau_{\max} - \sigma} 1 = n, \tag{5.19}$$

$$\sum_{\tau_{ij} \leq \tau_{\min} + \sigma} 1 = |E| - n, \tag{5.20}$$

onde σ é um limiar de imprecisão, n é o tamanho das soluções e $|E|$ é a quantidade de arestas. A convergência ocorre quando n arestas atingem o limitante superior do valor do feromônio e quando as outras $|E| - n$ arestas restantes atingem o limitante inferior do valor do feromônio. Uma vez que o sistema encontra-se nessa situação, definida pelas Equações 5.19 e 5.20, 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 dessa reinicialização do procedimento, os valores atuais de $F(S_{melhor})$ e $F(S_{pior})$ são mantidos, fazendo com que os valores calculados pela Equação 5.18 sejam menores do que na condição inicial.

São essas as três propostas que serviram de base para a implementação do SwarmBcluster, com exceção da detecção de convergência do MMAS melhorado. O SwarmBcluster tem uma maneira própria de lidar com a convergência.

De maneira geral, tanto no SwarmBcluster, como no ACO original, a proposta de cada formiga é descobrir a sequência ótima de vértices a visitar. Como o SwarmBcluster objetiva encontrar biclusters, inicialmente, as formigas começarão com um bicluster composto por uma linha da matriz, a qual representa o primeiro vértice visitado. Então, num processo iterativo, as formigas vão escolhendo as outras linhas que irão compor o bicluster. Antes de descrever o algoritmo SwarmBcluster em detalhes, a heurística construtiva para a busca de δ-biclusters utilizada por ele será descrita no Algoritmo 5.4 e explicada a seguir.

Para cada linha que é adicionada ao bicluster que está sendo formado, o MSR de cada coluna do bicluster é calculado. As colunas que contribuem para um MSR maior que δ são excluídas segundo a ordem de participação no MSR. As colunas que mais contribuem para o

MSR daquele bicluster são excluídas primeiro, até se obter MSR menor que δ. Se o bicluster resultante deste processo possui menos colunas que *col_min*, então o bicluster volta ao estado anterior e, em seguida, a mesma linha é adicionada com valores negativos. Se mesmo assim o bicluster resultante desse processo possui menos colunas que *col_min*, então ele volta novamente ao estado anterior. E tenta-se adicionar outra linha. Após adicionar todas as linhas possíveis ao bicluster, ocorre um processo de busca local, chamado *Node Addition*, descrito em CHENG & CHURCH (2000). Se possível, essa busca local aumentará o volume do bicluster. Isso acontece porque 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. O mesmo vale para as colunas, ou seja, uma coluna que não era apta a ser adicionada ao bicluster, em determinado momento, pode se tornar apta com o conjunto de linhas do bicluster final. Se o bicluster resultante desse processo possuir menos linhas que *row_min*, ele será desconsiderado e uma nova busca pode ser realizada com uma nova linha como ponto inicial.

Algoritmo 5.4: [lista de biclusters] = Heurística (δ, col_min, row_min)

```
% col_min – quantidade mínima de colunas em um bicluster
% row_min – quantidade mínima de linhas em um 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 MSR de cada coluna segundo a Eq. 4.12
    Remova todas as colunas que possuírem MSR 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 MSR de cada coluna, segundo a Eq. 4.12
        Remova todas as colunas que possuírem MSR 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 a busca local chamada Node Addition (CHENG & CHURCH, 2000)
Se o bicluster resultante possuir menos linhas que row_min então
    Descarte o bicluster
Fim Se
```

Para exemplificar o processo de construção de um bicluster através da heurística apresentada no Algoritmo 5.4, será mostrado como encontrar um bicluster com $\delta = 1$ na matriz de dados apresentada na Figura 5.2. A ordem escolhida para inserção das linhas no bicluster é 1, 2, 3 e 4.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	2	3	5	1	2	4	5	3	2	1	1	2
2	2	1	1	4	1	3	4	4	5	1	2	3	2	1
3	3	3	4	5	1	3	5	2	1	5	4	3	3	4
4	3	5	3	2	4	4	3	3	4	2	5	4	4	5

Figura 5.2: Exemplo de uma matriz de dados.

Então, o processo de construção inicia com as linhas 1 e 2. Após a adição dessas duas linhas ao bicluster, é necessário calcular o MSR de cada coluna e, em seguida, remover as colunas que violam a restrição imposta por δ, conforme exibido na Figura 5.3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	2	3	5	1	2	4	5	3	2	1	1	2
2	2	1	1	4	1	3	4	4	5	1	2	3	2	1
	0,2	0	0,3	0,2	4,1	0,9	0,9	0	0	1,1	0	0,9	0,2	0,3

Figura 5.3: Primeira etapa do exemplo de construção de um bicluster a partir da matriz de dados apresentada na Figura 5.2.

Após remover as colunas 5 e 10 do bicluster, a terceira linha deve ser adicionada e, novamente, é necessário remover as colunas com MSR > 1, conforme exibido na Figura 5.4.

	1	2	3	4	6	7	8	9	11	12	13	14
1	1	1	2	3	1	2	4	5	2	1	1	2
2	2	1	1	4	3	4	4	5	2	3	2	1
3	3	3	4	5	3	5	2	1	4	3	3	4
	0,1	0,3	0,9	0,1	0,3	0,6	2	5,5	0,3	0,3	0,1	0,9

Figura 5.4: Segunda etapa do exemplo de construção de um bicluster a partir da matriz de dados apresentada na Figura 5.2.

Após remover as colunas 8 e 9 do bicluster, a quarta linha deve ser adicionada e, novamente, é necessário remover as colunas com MSR > 1, conforme exibido na Figura 5.5.

	1	2	3	4	6	7	11	12	13	14
1	1	1	2	3	1	2	2	1	1	2
2	2	1	1	4	3	4	2	3	2	1
3	3	3	4	5	3	5	4	3	3	4
4	3	5	3	2	4	3	5	4	4	5
	0	0,9	0,5	2	0,3	0,9	0,3	0,3	0,1	0,8

Figura 5.5: Terceira etapa do exemplo de construção de um bicluster a partir da matriz de dados apresentada na Figura 5.2.

O bicluster resultante é exibido na Figura 5.6. Ele possui MSR = 0,45 e volume = 36.

	1	2	3	6	7	11	12	13	14
1	1	1	2	1	2	2	1	1	2
2	2	1	1	3	4	2	3	2	1
3	3	3	4	3	5	4	3	3	4
4	3	5	3	4	3	5	4	4	5

Figura 5.6: Exemplo de bicluster resultante da matriz de dados apresentada na Figura 5.2.

É fácil perceber que a qualidade de um bicluster construído pela heurística descrita no Algoritmo 5.4 é influenciada pela ordem em que as linhas são inseridas no bicluster. Por exemplo, considerando a ordem de inserção das linhas: 4, 3, 2 e 1, o bicluster resultante da matriz de dados da Figura 5.2 teria: as colunas 1, 2, 3, 6, 11, 12, 13 e 14; MSR 0,33; e volume 32. Por isso que no SwarmBcluster, assim como no ACO para TSP, a proposta de cada formiga é descobrir a sequência ótima de vértices a visitar.

Algoritmo 5.5: [lista de biclusters] = SwarmBcluster(bic_max, over_thr)

```
% bic_max – quantidade máxima de biclusters que se deseja encontrar
% over_thr – taxa máxima de sobreposição entre dois biclusters
Crie uma lista de biclusters vazia
Inicialize a lista de candidatos com as M linhas e N colunas da matriz de
dados, como exemplificado na Figura 5.7
Enquanto a lista de candidatos não estiver vazia E a lista de biclusters é
menor que bic_max faça
    Pegue um candidato da lista de candidatos para ser o ponto inicial
    Execute o algoritmo ACO, utilizando a heurística construtiva, com esse
ponto inicial
    Se o bicluster resultante sobrepõe qualquer outro da lista de biclusters
em mais de over_thr% então
        Se o bicluster atual é melhor que o bicluster sobreposto então
            Troque o bicluster sobreposto
        Fim Se
    Senão
        Insira o bicluster na lista de biclusters
    Fim Se
    Atualize a lista de candidatos removendo os elementos contidos no
bicluster gerado
Fim Enquanto
```

Para o SwarmBcluster, esses vértices são as linhas da matriz de dados que está sendo analisada. Sendo assim, o SwarmBcluster, basicamente, aplica o algoritmo ACO (descrito no Algoritmo 5.3 e com as melhorias propostas) utilizando essa heurística construtiva. O Algoritmo 5.5 descreve o SwarmBcluster.

A *lista de biclusters* é a estrutura que conterá todos os biclusters encontrados dentro da base de dados que está sendo analisada. Ela também lidará com a sobreposição dos biclusters, pois nesta lista não haverá nenhum bicluster que sobreponha algum outro da lista em mais de *over_thr* por cento.

A *lista de candidatos* é a estrutura que irá guiar as formigas a procurarem biclusters em diferentes regiões da base de dados, lidando, portanto, com a cobertura da base de dados. Ela inicia com uma lista de todos os índices de linhas da base de dados, contendo uma sub-lista de todos os índices de colunas, conforme exemplo da Figura 5.7. Para executar o algoritmo ACO, é escolhida uma linha desta *lista de candidatos*, a qual será o ponto de partida das formigas. A linha escolhida será aquela com maior número de índices de colunas. É por isso que esta estrutura guia a busca de biclusters nas regiões não-exploradas da base de dados.

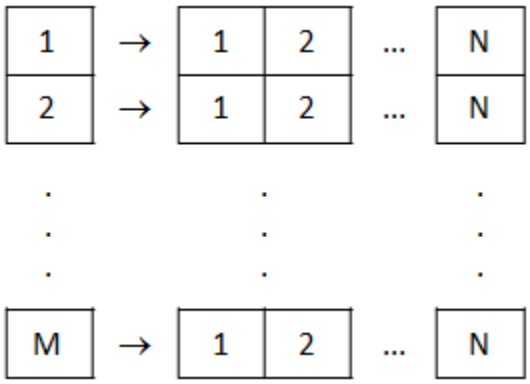


Figura 5.7: Lista de candidatos inicializada para uma matriz de *M* linhas e *N* colunas.

Para encontrar a sequência ótima de linhas que vão maximizar o volume do bicluster, respeitando o MSR máximo admitido, aqui definido como δ, as formigas escolhem a próxima linha a ser adicionada no bicluster de acordo com a Equação 5.14, sendo η_{ij} igual ao inverso da distância euclidiana entre a linha *i* e *j* da base de dados. As linhas com maior probabilidade, p_{ij}, têm mais chances de serem adicionadas ao bicluster.

Quando todas as formigas terminam de construir seus biclusters, um para cada uma, a matriz de feromônio é atualizada com base na formiga que produziu o melhor bicluster, através das Equações 5.15 e 5.18.

O algoritmo ACO irá retornar a melhor solução encontrada, nesse caso, o melhor bicluster encontrado, isto é, o bicluster de maior volume e que respeita a restrição de MSR (δ). Se este bicluster encontrado não sobrepuser nenhum outro da *lista de biclusters* em mais de *over_thr* por cento, então ele é adicionado automaticamente à lista. Caso contrário, é averiguado se ele possui maior volume que o bicluster sobreposto e, em caso positivo, o bicluster sobreposto é trocado pelo recentemente descoberto.

Para finalizar, os elementos (linhas e colunas) que formaram o novo bicluster são removidos da lista de candidatos. Se uma determinada entrada da lista de candidatos se tornar um conjunto vazio durante esse processo, ela é removida da lista. Com este procedimento, as novas buscas irão acontecer em uma região diferente da base de dados. A Figura 5.8 ilustra uma lista de candidatos que já foi atualizada após a formação de biclusters, ou seja, ela mostra as linhas / colunas da base de dados que ainda não foram cobertas por biclusters.

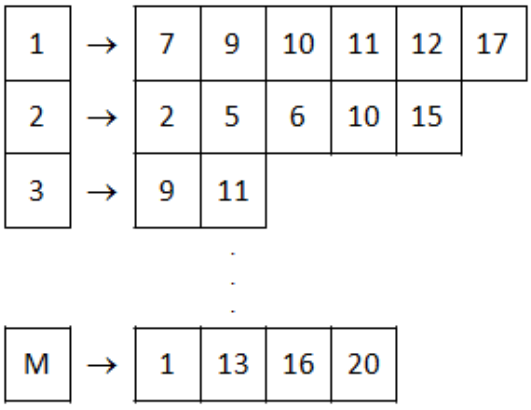


Figura 5.8: Lista de candidatos representando as colunas ainda não-cobertas pelos biclusters já produzidos.

A próxima subseção irá descrever o algoritmo SwarmBcluster especializado para a imputação de dados faltantes.

5.3.1 SwarmBcluster para Imputação de Dados Faltantes

Para que o SwarmBcluster trabalhe com a imputação dos dados faltantes, DE FRANÇA (2010) propôs algumas modificações no SwarmBcluster original (DE FRANÇA & VON ZUBEN, 2010), cujo objetivo era apenas encontrar biclusters em uma base de dados.

A primeira delas se refere à cobertura da base de dados. Na versão original, deseja-se que os biclusters cubram toda a base de dados. Na versão para imputação, deseja-se, principalmente, que os biclusters cubram todos os dados faltantes. Para isso, além da *lista de candidatos* original, que representa a lista de elementos que ainda não foram cobertos até o momento, existe também uma *lista de dados faltantes*.

A *lista de dados faltantes* é apenas uma matriz bidimensional, na qual cada linha fornece a posição de um dado faltante na matriz que está sendo analisada. Ela é inicializada com uma lista contendo todas as posições dos dados faltantes existentes na base de dados. A Figura 5.9 traz um exemplo, no qual a matriz que está sendo analisada possui 3 dados faltantes. O primeiro está localizado na linha 3, coluna 10; o segundo está localizado na linha 1098, coluna 32; e o terceiro está localizado na linha 1890, coluna 5.

A segunda adaptação do SwarmBcluster se refere ao grau de sobreposição dos biclusters. Na versão original, os biclusters produzidos não sobrepõem um ao outro em mais de *over_thr* por cento (um parâmetro determinado pelo usuário). Na versão para imputação, essa restrição não existe. Logo, com o passar das iterações, os valores imputados por um novo bicluster têm uma maior chance de já terem sido imputados por outro bicluster encontrando anteriormente. Com a restrição de sobreposição, poderia ser necessário um número menor de biclusters para cobrir todo o conjunto de dados, porém, isso também restringiria o volume dos biclusters (DE FRANÇA, 2010).

3	10
1098	32
1890	5

Figura 5.9: Exemplo de lista de dados faltantes.

Outra dificuldade encontrada por essa adaptação do SwarmBcluster é como encontrar biclusters em uma base de dados incompleta. Segundo DE FRANÇA (2010), as técnicas de

biclusterização apresentam robustez quanto ao ruído dentro da base de dados. Portanto, o autor optou por realizar uma pré-imputação na base de dados utilizando uma técnica simples, produzindo, geralmente, uma base de dados ruidosa. Desse modo, através dos biclusters coerentes encontrados 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. A técnica escolhida foi o *vizinho mais próximo* (registro mais similar, como explicado na Subseção 3.4.2), utilizando a distância euclidiana como métrica de similaridade. Quando não for possível encontrar um vizinho mais próximo, utiliza-se a mediana.

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & \_ & 7 & 9 \\ 5 & 7 & 9 & 11 \\ 7 & 9 & \_ & 13 \end{bmatrix} \qquad \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & x_1 & 7 & 9 \\ 5 & 7 & 9 & 11 \\ 7 & 9 & x_2 & 13 \end{bmatrix}$$

(a) (b)

Figura 5.10: Exemplo de um bicluster coerente com dados faltantes (a) sendo convertidos em variáveis a serem otimizadas (b).

Para encontrar tais valores, mais próximos aos valores reais, os dados pré-imputados presentes nos biclusters gerados são reimputados. Para isso, os dados pré-imputados são convertidos em variáveis que devem ser obtidas de modo a minimizar o MSR. A Figura 5.10 traz um exemplo de um bicluster coerente com dados faltantes (5.10a), sendo convertidos em variáveis a serem otimizadas (5.10b).

Como o problema de minimizar o MSR pode ser modelado como um problema de otimização quadrática, a seguinte descrição foi utilizada (DE FRANÇA, 2010):

$$\begin{aligned} &\text{Min } H_{IJ}(x) = \frac{1}{2} x^T Q x + b^T x + c \\ &\text{s.a. } lb_i \le x_i \le ub_i, i = 1, ..., n \end{aligned} \tag{5.21}$$

onde $x = [x_1, x_2, ..., x_n]^T$, n é a quantidade de dados faltantes presentes dentro do bicluster, $Q \in \Re^{nxn}$, $b \in \Re^n$, $c \in \Re$, e lb_i e ub_i são os limites inferiores e superiores de cada variável x_i, respectivamente. Esse problema de otimização quadrática pode ser resolvido por vários algoritmos presentes na literatura. Mas, para isso, é necessário conhecer os valores de Q, b e c.

Para calcular esses valores, deve-se recorrer à Equação 4.12, a qual calcula o MSR e será relembrada aqui:

$$H_{IJ}(x) = \frac{1}{|I||J|}\sum_{i,j\in I,J} r_{ij}^2(x) = \frac{1}{|I||J|}\sum_{i,j\in I,J}(a_{ij}(x) - a_{Ij}(x) - a_{iJ}(x) + a_{IJ}(x))^2, \quad (5.22)$$

onde $H_{IJ}(x)$ é o MSR do bicluster $A(I, J)$, considerando o vetor de variáveis x, e $|\zeta|$ é a quantidade de elementos do conjunto ζ. A matriz Q é obtida calculando a hessiana de $H_{IJ}(x)$. O vetor b é o vetor gradiente de $H_{IJ}(x)$ quando $x = 0$. E o valor de c é obtido através do cálculo de $H_{IJ}(0)$.

O gradiente da função quadrática (Equação 5.21) é dado por:

$$\nabla H_{IJ}(x) = Qx + b, \quad (5.23)$$

com $\nabla H_{IJ} \in \Re^n$. Logo, considerando a Equação 5.22, que calcula o MSR, tem-se:

$$\nabla_k H_{IJ}(x) = \frac{2}{|I||J|}\cdot\left[\sum_{i,j\in I,J} r_{ij}(x)\cdot\frac{\partial r_{ij}(x)}{\partial x_k}\right], \quad (5.24)$$

onde $\nabla_k H_{IJ}$ denota o k-ésimo elemento de ∇H_{IJ}, com $k = 1, ..., n$. Então, pode-se calcular cada elemento do vetor b através de:

$$b_k = \nabla_k H_{IJ}(0) = \frac{2}{|I||J|}\cdot\left[\sum_{i,j\in I,J} r_{ij}(0)\cdot\frac{\partial r_{ij}(0)}{\partial x_k}\right]. \quad (5.25)$$

A matriz Q é obtida através da Hessiana da função quadrática (Equação 5.21):

$$Q = \nabla^2 H_{IJ}(x). \quad (5.26)$$

Desse modo, considerando a Equação 5.22, que calcula o MSR, tem-se:

$$q_{kl} = \nabla^2_{kl} H_{IJ}(x) = \frac{2}{|I||J|}\cdot\sum_{i,j\in I,J}\left[\frac{\partial r_{ij}(x)}{\partial x_l}\cdot\frac{\partial r_{ij}(x)}{\partial x_k}\right], \quad (5.27)$$

onde q_{kl} denota o elemento da linha k e coluna l da matriz Q.

Percebe-se que para se obter Q e b é necessário calcular as 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 de cada variável:

$$\frac{\partial r_{ij}}{\partial x_k} = \begin{cases} c_1 = 1 - \frac{1}{|I|} - \frac{1}{|J|} + \frac{1}{|I \| J|}, \text{se } i = k_i, j = k_j \\ c_2 = -\frac{1}{|J|} + \frac{1}{|I \| J|}, \text{se } i = k_i, j \neq k_j \\ c_3 = -\frac{1}{|I|} + \frac{1}{|I \| J|}, \text{se } i \neq k_i, j = k_j \\ c_4 = \frac{1}{|I \| J|}, \text{se } i \neq k_i, j \neq k_j \end{cases}, \quad (5.28)$$

onde k_i e k_j dizem respeito à posição da linha e da coluna, respectivamente, da variável k no bicluster.

Com algumas simplificações, as Equações 5.25 e 5.27 se tornam (DE FRANÇA, 2010):

$$b_k = \nabla_k H_{IJ}(0) = \frac{2}{|I \| J|} \cdot \left[a_{IJ} - a_{k_i J} - a_{I k_j} \right], \quad (5.29)$$

$$q_{kl} = \nabla^2_{kl} H_{IJ}(x) = \frac{2}{|I \| J|} \cdot \frac{\partial r_{k_i k_j}}{\partial x_l}. \quad (5.30)$$

Como já foi dito, através dos valores de Q e b, os valores das variáveis (dados faltantes) presentes dentro de um bicluster podem ser encontrados facilmente através de algoritmos convencionais de otimização quadrática. Mas, para a solução desse problema em tempo polinomial, deve-se garantir que a matriz Q seja definida positiva, para que exista a inversa da matriz Q e um ótimo global finito. Conforme demonstrado por DE FRANÇA (2010), a taxa máxima de valores ausentes, r, deve ser:

$$r < \frac{|I \| J| - 2}{3|I \| J| - 2|I| - 2|J|} \quad (5.31)$$

A versão do SwarmBcluster para imputação está descrita no Algoritmo 5.6 e será explicada a seguir.

A lista de dados faltantes irá guiar as formigas a procurarem biclusters nas regiões da base de dados que possuem dados faltantes, lidando, portanto, com a cobertura desses. Para iniciar o algoritmo ACO, é escolhida, da lista de candidatos, a linha correspondente ao dado faltante. Por exemplo, suponha que um dos elementos da lista de dados faltantes seja (3, 5). Isso significa que existe um dado faltante na linha 3 e coluna 5 da base de dados. Nesse caso, a linha 3 da lista de candidatos seria o ponto de partida das formigas no algoritmo ACO. Se a linha 3 da lista de candidatos possuir uma quantidade de colunas inferior a *col_min*, o ACO é iniciado com a linha 3

e todas as colunas da matriz de dados. Se o objetivo do algoritmo ACO é encontrar um bicluster que cubra o dado faltante em questão, não faz sentido permitir que ele remova a coluna do dado faltante. Nesse exemplo, não seria permitida a remoção da coluna 5.

Algoritmo 5.6: [IMP] = SwarmBcluster(max_vars)

```
% max_vars – quantidade máxima de dados faltantes presentes em um bicluster
Impute os dados faltantes utilizando uma metodologia simples
Crie a lista de candidatos
Crie a lista de dados faltantes
Para cada dado faltante na lista de dados faltantes faça
    Selecione a linha da lista de candidatos correspondente à posição do dado faltante
    Aplique o algoritmo ACO com essa linha inicial, impedindo a remoção da coluna que contém o dado faltante
    Se Bicluster gerado atende as restrições de taxa máxima de dados faltantes então
        Aplique um método de otimização quadrática e guarde os valores gerados, sem imputá-los
    Senão
        Armazene o bicluster em memória
    Fim Se
    Atualize a lista de candidatos de acordo com as linhas e colunas cobertas
    Atualize a lista de dados faltantes de acordo com as variáveis ausentes calculadas
Fim Para
Impute valores já calculados na base de dados
Processe novamente biclusters armazenados em memória
```

Ao encontrar o bicluster, ele é avaliado quanto à restrição de *row_min*. Se o bicluster possuir menos linhas que *row_min*, ele é descartado. Nesse caso, não foi possível encontrar um bicluster de volume aceitável que contivesse o dado faltante e, portanto, esse dado faltante permanece com o valor pré-imputado.

O bicluster gerado também deve atender a restrição de quantidade máxima de valores faltantes, o que é testado através da Equação 5.31 e do parâmetro *max_vars*. O parâmetro *max_vars* é utilizado porque uma quantidade muito grande de dados faltantes torna a otimização muito complicada e custosa. Se a quantidade máxima de valores faltantes for respeitada, utiliza-se um algoritmo de otimização quadrática para encontrar os valores dos dados faltantes. Caso contrário, se a quantidade máxima de valores faltantes não for respeitada, o bicluster é armazenado em memória para uso posterior. O algoritmo de otimização quadrática utilizado por DE FRANÇA (2010) é o método proposto em GOLDFARB & IDNANI (1983).

Como pode ocorrer de um mesmo dado faltante estar presente em mais de um bicluster, os valores faltantes já estimados são armazenados para serem imputados somente no final do processo. Assim, se isso acontecer, o valor imputado será a média ponderada dos valores calculados. O peso dessa média é a razão entre o volume do bicluster e o número de valores ausentes dentro dele.

Após a construção e validação do bicluster, a lista de candidatos e a lista de dados faltantes são atualizadas. É importante enfatizar que, quando se inicia a construção de um bicluster, ele objetiva cobrir pelo menos um dado faltante, mas pode ser que o bicluster resultante cubra diversos dados faltantes. Portanto, não é necessário procurar novos biclusters para tais dados faltantes também cobertos.

Quando toda a lista de dados faltantes termina de ser percorrida, é realizada a imputação dos valores já calculados previamente. Após a imputação de todos os dados faltantes estimados, os biclusters que não respeitavam a quantidade máxima de valores ausentes são reprocessados, dessa vez, levando em conta os valores já imputados. Desse modo, a matriz Q que não era definida positiva, pode passar a ser. Assim, os últimos dados faltantes são imputados e o procedimento termina. Se não foi possível encontrar biclusters que cobrissem todos os dados faltantes, ou se alguma matriz Q não se tornou definida positiva, tais dados faltantes continuam com os valores pré-imputados.

5.4 Síntese do Capítulo

Este capítulo descreveu os três algoritmos que foram utilizados nos experimentos a serem apresentados no Capítulo 6.

O primeiro deles foi o algoritmo KNNImpute. A descrição desse algoritmo é baseada no algoritmo de *K-Vizinhos mais próximos* para imputação de dados faltantes, proposto em CANDILLIER (2008). O segundo deles foi o rSVD, um algoritmo baseado em *Decomposição de Valores Singulares com regularização*, criado por FUNK (2006). E, finalmente, o terceiro algoritmo foi o SwarmBcluster, um algoritmo de biclusterização inspirado em Inteligência de Enxames, que foi adaptado para a imputação de dados faltantes por DE FRANÇA (2010).

O algoritmo SwarmBcluster também será utilizado no Capítulo 7, no qual serão exibidos os resultados dos experimentos envolvendo biclusterização e imputação múltipla. Como o algoritmo

SwarmBcluster trabalha de forma estocástica, as $x \geq 2$ imputações para cada dado faltante são obtidas, simplesmente, através de x execuções do algoritmo com diferentes sementes aleatórias.

Capítulo 6

Análise de sensibilidade paramétrica e comparações para imputação única

Para avaliar a eficácia das abordagens propostas no Capítulo 5, foram escolhidas duas bases de dados de expressão gênica e uma base de dados experimental de filtragem colaborativa.

Essas três bases de dados possuem dados faltantes, porém eles foram eliminados para que se pudesse trabalhar com três bases de dados completas. Um conjunto de dados completo deve ser considerado como ponto de partida, seguido pela introdução artificial de dados faltantes, porque, dessa forma, uma comparação adequada entre os valores reais e os estimados pode ser realizada. A partir de uma única base de dados completa, foram criadas várias, variando o mecanismo e a quantidade de dados faltantes.

A qualidade dos valores imputados foi avaliada através do erro absoluto médio (MAE – do inglês *Mean Absolute Error*) e da raiz quadrada do erro quadrático médio (RMSE – do inglês *Root Mean Square Error*), respectivamente:

$$MAE = \frac{\sum_{i=1}^{N} |x_i - \hat{x}_i|}{N}, \tag{6.1}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}, \tag{6.2}$$

onde x_i é o i-ésimo valor real da base de dados completa, $\hat{x}_i$ é o i-ésimo valor imputado e N é a quantidade de valores estimados. Quanto menor o valor de MAE e RMSE, melhor a precisão da imputação realizada. Basicamente, a diferença entre essas duas métricas é que MAE mostra a média dos erros absolutos, considerando, portanto, que qualquer tipo de erro tem o mesmo peso. Já o RMSE dá pesos maiores aos erros maiores, pela presença do termo quadrático.

A primeira base de dados é a *Saccharomyces cerevisiae cell cycle expression data* utilizada inicialmente por CHO *et al.* (1998) e mais conhecida como *Yeast*. Trata-se de um conjunto de

dados de expressão gênica, contendo 2.884 genes sob 17 condições, que pode ser encontrada em http://arep.med.harvard.edu/biclustering. Essa base possui dois registros faltantes, portanto este trabalho utiliza apenas os 2.882 registros completos que ela possui. A partir dessa base de dados com 2882 linhas e 17 colunas, foram criadas 31 bases de dados incompletas, variando o mecanismo e a quantidade de dados faltantes.

A segunda base de dados é a *Human B-Cell Lymphoma expression data*, mais conhecida como *Human* e utilizada pela primeira vez por ALIZADEH *et al*. (2000). Ela possui 4.026 genes sob 96 condições e também pode ser encontrada em http://arep.med.harvard.edu/biclustering. Essa base possui cerca de 5% de dados faltantes, os quais são representados pelo valor *999*. Portanto, para este trabalho, foram retiradas todas as linhas e colunas com dados faltantes, resultando uma matriz com 3.300 linhas e 40 colunas. A partir dessa matriz, novamente foram criadas 31 bases de dados incompletas.

A terceira base de dados é a *Jester*, que é utilizada para testar algoritmos de filtragem colaborativa. Ela foi utilizada pela primeira vez por GOLDBERG *et al*. (2001). Trata-se de um conjunto de dados que armazena a opinião de 24.983 usuários sobre 101 piadas. A aplicação web pode ser vista em http://eigentaste.berkeley.edu/user/index.php e a base de dados pode ser encontrada em http://goldberg.berkeley.edu/jester-data/. Retirando todos os dados faltantes, foi produzida uma matriz com 7.199 linhas, que representam os usuários, e 58 colunas, que representam as piadas. A partir dessa matriz, foram criadas 6 bases de dados incompletas, utilizando o mecanismo MCAR.

Conforme mencionado anteriormente, as três bases de dados utilizadas (Yeast, Human e Jester) são simplesmente matrizes de dados, em que as linhas representam objetos e as colunas representam atributos. Para gerar uma base (matriz) de dados incompleta segundo o mecanismo MCAR, uma quantidade de células foi escolhida aleatoriamente para ser ausente. Por exemplo, seja essa quantidade 2%, então, 2% das células da matriz foram escolhidas aleatoriamente para serem ausentes. Para gerar uma matriz de dados incompleta segundo o mecanismo MAR, um atributo foi escolhido para ser o motivo da ausência de dados e outros quatro atributos foram escolhidos para possuírem dados faltantes. Esses quatro atributos tinham certa probabilidade de serem ausentes sempre que o valor do atributo motivo fosse maior que sua média. Para gerar uma matriz de dados incompleta segundo o mecanismo MNAR, um atributo foi escolhido para ser o

motivo da ausência de dados. Esse atributo tinha certa probabilidade de ser ausente quando o seu valor era maior que a sua média.

Para as bases de dados Yeast e Human, os experimentos listados a seguir estão organizados da seguinte forma. Os três primeiros experimentos estão relacionados à análise de sensibilidade paramétrica. Eles avaliam o desempenho do SwarmBcluster com relação à variação dos seguintes parâmetros: δ, max_it e n_ants. A análise de sensibilidade paramétrica foi realizada com bases de dados que possuem dados faltantes segundo o mecanismo MCAR. O quarto experimento avalia o desempenho da heurística construtiva utilizada pelo SwarmBcluster. Para finalizar os experimentos com o mecanismo MCAR, o quinto experimento compara o desempenho dos três algoritmos descritos no Capítulo 5 com relação ao mecanismo MCAR. O sexto e o sétimo experimentos comparam o desempenho dos três algoritmos descritos no Capítulo 5 com relação aos mecanismos MAR e MNAR, respectivamente.

Para a base de dados Jester foram realizados apenas dois experimentos. O primeiro experimento avalia o desempenho do SwarmBcluster com relação à variação de δ. O segundo experimento compara o desempenho dos três algoritmos descritos no Capítulo 5. Todos os experimentos realizados com a Jester foram feitos utilizando o mecanismo MCAR. O número menor de experimentos se justifica por dois motivos: (*i*) os testes realizados com as bases de dados Yeast e Human mostraram que o parâmetro que exerce maior influência sobre a qualidade da imputação é o δ e (*ii*) como será visto na Seção 6.3, os testes realizados com o SwarmBcluster e a base de dados Jester possuem um custo computacional muito alto.

Também devido ao custo computacional do SwarmBcluster, todos os experimentos foram realizados com apenas uma execução desse algoritmo.

Os parâmetros δ, max_it e n_ants foram escolhidos para a análise de sensibilidade paramétrica porque esses são os parâmetros que possivelmente mais influenciam a qualidade dos biclusters gerados e, consequentemente, a qualidade das imputações. O valor de δ tem um grande impacto no compromisso entre a qualidade e o volume do bicluster. Quanto menor o valor de δ, menor será o volume do bicluster, em contrapartida, maior será a sua coerência. Mas, mesmo sendo mais coerente, esse bicluster pode não trazer muita informação para o processo de imputação devido ao baixo volume. Os parâmetros max_it e n_ants influenciam diretamente a qualidade das soluções encontradas pelo algoritmo ACO, o que consequentemente afeta a qualidade dos biclusters encontrados.

6.1 Base de dados Yeast

Nesta seção, serão descritos os experimentos realizados com a base de dados Yeast. Esses experimentos foram realizados em um computador AMD Athlon(tm) 64 X2 Dual Core Processor 4000 @ 2.11 Ghz, 1 GB de RAM, utilizando apenas um único *core* para cada experimento.

Os parâmetros-base para os experimentos que foram realizados são listados a seguir. O parâmetro utilizado para o KNNImpute foi $K = 20$. Como foi mencionado na Seção 5.1, para encontrar um valor de K apropriado para os experimentos deste trabalho, foram realizados alguns experimentos preliminares. Os parâmetros utilizados para o rSVD são apresentados na Tabela 6.1. A parametrização do rSVD foi baseada no trabalho de DE FRANÇA (2010). Os parâmetros utilizados para o SwarmBcluster são apresentados na Tabela 6.2. O valor de $\delta = 300$ foi determinado por CHENG & CHURCH (2000), baseado nos estudos de clusters realizados nessa base por TAVAZOIE *et al*. (1999). Os valores dos outros parâmetros foram baseados em DE FRANÇA (2010).

Tabela 6.1: Parâmetros utilizados pelo rSVD.

Parâmetro	**Valor**
F	5
max_it	120
min_improvement	10^{-4}
init_val	0,1
Lrate	10^{-5}
λ	0,015

Tabela 6.2: Parâmetros-base utilizados pelo SwarmBcluster nos experimentos com a base de dados Yeast.

Parâmetro	**Valor**
δ	300
max_it	5
n_ants	5
α	1
β	3
ρ	0,2
col_min	7
row_min	100
max_vars	2500

A análise de sensibilidade paramétrica exibida a seguir foi realizada com bases de dados que possuem dados faltantes segundo o mecanismo MCAR. Para realizar os experimentos de análise de sensibilidade paramétrica do algoritmo SwarmBcluster, foram escolhidas as bases de dados que possuem 15%, 50% e 80% de dados faltantes. Além de ser uma porcentagem pequena, uma média e uma alta de dados faltantes, elas foram escolhidas porque em DE FRANÇA (2010): (*i*) com 15%, o SwarmBcluster foi bastante estável e, além disso, apresentou seu melhor resultado; (*ii*) com 50%, a qualidade da imputação do SwarmBcluster começou a piorar; e (*iii*) com 80%, o SwarmBcluster obteve seu segundo pior resultado, mas ainda com possibilidade de melhora.

6.1.1 Primeiro Experimento: a influência de δ

O primeiro experimento investiga a influência da variação do parâmetro δ no algoritmo SwarmBcluster. Os parâmetros utilizados são apresentados na Tabela 6.2, com exceção de δ, o qual foi variado.

A influência da variação de δ no valor de MAE é mostrada na Tabela 6.3.

Para a base com 15% de dados faltantes, a diminuição no valor de δ trouxe resultados melhores para o valor de MAE e do desvio-padrão, enquanto o aumento no valor de δ trouxe resultados piores. Com δ = 200 e δ = 700, o resultado de MAE foi, respectivamente, 6,76% melhor e 19,48% pior que com δ = 300. O valor máximo do erro absoluto também apresentou correlação positiva com o valor de δ. A mediana se comportou de forma idêntica ao valor de MAE e do desvio-padrão, isto é, quanto maior o valor de δ, pior o valor da mediana. Para todos os valores de δ, foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Os testes com a base com 50% de dados faltantes possuíram um comportamento semelhante aos testes com a base de 15%, ou seja, valores menores de δ geraram melhores resultados. Com δ = 200 e δ = 700, o resultado de MAE foi, respectivamente, 5,23% melhor e 15,94% pior que com δ = 300. O valor máximo do erro absoluto não apresentou correlação com o valor de δ. A mediana e o desvio-padrão se comportaram da mesma forma que o valor de MAE. Para todos os valores de δ, foram obtidas imputações perfeitas. Logo, em pelo menos um caso o erro absoluto foi igual a zero. Vale observar que com um valor de δ = 100, o algoritmo não foi capaz de encontrar biclusters respeitando a restrição imposta pela Equação 5.31 para ser possível

convergir para um resultado factível no problema de otimização quadrática. Portanto, os valores para esse caso foram omitidos.

Ao contrário dos testes realizados com as bases de dados com 15% e 50% de dados faltantes, a base de dados com 80% apresentou melhores resultados para MAE e desvio-padrão com $\delta > 300$. O melhor valor de δ foi 400, sendo que nesse experimento o valor de MAE foi 14,29% melhor que com $\delta = 300$. Para $\delta > 400$, os resultados pioraram se comparados a $\delta = 400$, mas continuaram melhores que com $\delta = 300$. O valor máximo do erro absoluto não apresentou correlação com o valor de δ. A mediana apresentou o mesmo comportamento que o apresentado pelo valor de MAE. Como nos testes com as bases com 15% e 50% de dados faltantes, para todos os valores de δ foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero. Novamente, com $\delta = 100$ o algoritmo não conseguiu encontrar biclusters que respeitavam o critério de convergência da otimização quadrática.

A influência da variação de δ no valor de RMSE é mostrada na Tabela 6.4.

Para a base com 15% de dados faltantes, a diminuição no valor de δ trouxe resultados melhores para o valor de RMSE e do desvio-padrão, enquanto o aumento no valor de δ trouxe resultados piores. Com $\delta = 200$ e $\delta = 700$, o resultado de RMSE foi, respectivamente, 5,64% melhor e 17,47% pior que com $\delta = 300$. Porém, o teste com $\delta = 100$ mostra que se o valor de δ for muito pequeno, os resultados irão piorar. Isso acontece porque fica impossível encontrar biclusters que cubram todos os dados faltantes. Logo, esses dados são preenchidos com os valores da pré-imputação.

Os testes com a base com 50% de dados faltantes apresentaram um comportamento semelhante aos testes com a base com 15%. Com $\delta = 200$ e $\delta = 700$, o resultado de RMSE foi, respectivamente, 3,10% melhor e 14,28% pior que com $\delta = 300$.

Ao contrário dos experimentos realizados com as bases de dados de 15% e 50% de dados faltantes, a base de dados com 80% apresentou melhores resultados para RMSE e desvio-padrão com $\delta > 300$. O melhor valor de δ foi 400, sendo que nesse experimento o valor de RMSE foi 12,49% melhor que com $\delta = 300$. Para $\delta > 400$, os resultados ainda foram melhores que o caso $\delta = 300$, mas com um percentual de melhora gradativamente menor.

Tabela 6.3: Influência da variação de δ no valor de MAE para a base Yeast (MCAR).

	15%						50%						80%					
δ	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
100	15,963	16,178	229,521	11,208	0,00	9,33												
200	16,416	15,620	156,874	11,891	0,00	6,76	18,525	18,779	270,134	13,007	0,00	5,23	52,237	68,486	595,000	27,537	0,00	-49,46
300	17,606	16,332	160,803	12,997	0,00	0,00	19,547	18,945	354,161	14,236	0,00	0,00	34,950	50,211	581,113	20,438	0,00	0,00
400	18,614	17,019	163,233	13,800	0,00	-5,72	20,323	19,628	416,000	14,996	0,00	-3,97	29,956	44,371	584,997	18,408	0,00	14,29
500	19,322	17,486	160,067	14,480	0,00	-9,75	20,903	20,018	335,932	15,438	0,00	-6,94	29,992	44,965	589,671	18,454	0,00	14,19
600	20,206	18,273	182,701	15,030	0,00	-14,77	21,672	20,307	314,000	16,240	0,00	-10,87	31,149	46,530	595,000	19,227	0,00	10,88
700	21,035	18,796	189,703	15,944	0,00	-19,48	22,662	21,312	308,260	16,895	0,00	-15,94	33,536	50,175	585,676	20,312	0,00	4,05

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com δ = 300.

Tabela 6.4: Influência da variação de δ no valor de RMSE para a base Yeast (MCAR).

	15%			50%			80%		
δ	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
100	22,728	35,233	5,36						
200	22,660	33,017	5,64	26,378	43,122	3,10	86,134	142,302	-40,79
300	24,015	34,412	0,00	27,221	44,440	0,00	61,177	121,253	0,00
400	25,221	36,011	-5,02	28,254	47,538	-3,79	53,536	117,323	12,49
500	26,060	37,146	-8,52	28,942	47,080	-6,32	54,050	120,939	11,65
600	27,243	39,286	-13,44	29,700	46,298	-9,11	55,994	127,384	8,47
700	28,209	39,897	-17,47	31,109	48,015	-14,28	60,351	130,859	1,35

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com δ = 300.

Tabela 6.5: Influência da variação de δ no tempo de execução para a base Yeast (MCAR).

	15%		50%		80%	
δ	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**
100	11,66	-683,87				
200	3,86	-159,30	9,24	-58,31	9,90	-13,12
300	1,49	0,00	5,84	0,00	8,75	0,00
400	0,97	34,83	7,89	-35,14	13,21	-50,94
500	0,91	38,85	11,34	-94,34	17,95	-105,09
600	1,00	32,67	13,59	-132,82	26,21	-199,47
700	1,64	-10,07	17,39	-197,95	41,65	-375,75

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com δ = 300.

A influência da variação de δ no tempo de execução é mostrada na Tabela 6.5. Para a base com 15% de dados faltantes, com δ até 500, quanto maior o valor de δ, menor o tempo de execução. Para δ = 600 e δ = 700, o tempo de execução é um pouco maior que com δ = 500. Para a base com 50% de dados faltantes, a variação de δ, para baixo ou para cima, só piorou o tempo de execução. O mesmo aconteceu com o caso de 80%.

A Figura 6.1 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução, com a variação no valor de δ. Fica claro que, com relação ao MAE e ao RMSE, os experimentos realizados com as bases de dados com 15% e 50% de dados faltantes possuíram um comportamento semelhante. Ambos, MAE e RMSE, pioraram com o aumento no valor de δ. Porém, a influência causada por δ nos valores de MAE e RMSE para o caso de 80% foi bem maior, além de diferente. Tanto os valores de MAE como os valores de RMSE melhoram significativamente com o crescimento de δ até 400. Isso pode ser explicado pelo fato de que,

quanto maior o valor de δ, maior é o volume esperado dos biclusters resultantes. E isso indica que, com altas porcentagens de dados faltantes, é mais desejável favorecer a maximização do volume dos biclusters gerados.

Em relação ao custo computacional, os experimentos com as bases de dados com 50% e 80% de dados faltantes possuíram um comportamento mais semelhante. Ambos foram muito sensíveis à variação de δ. A base de dados com 15% apresentou um comportamento bastante estável, principalmente com δ no intervalo [300, 700].

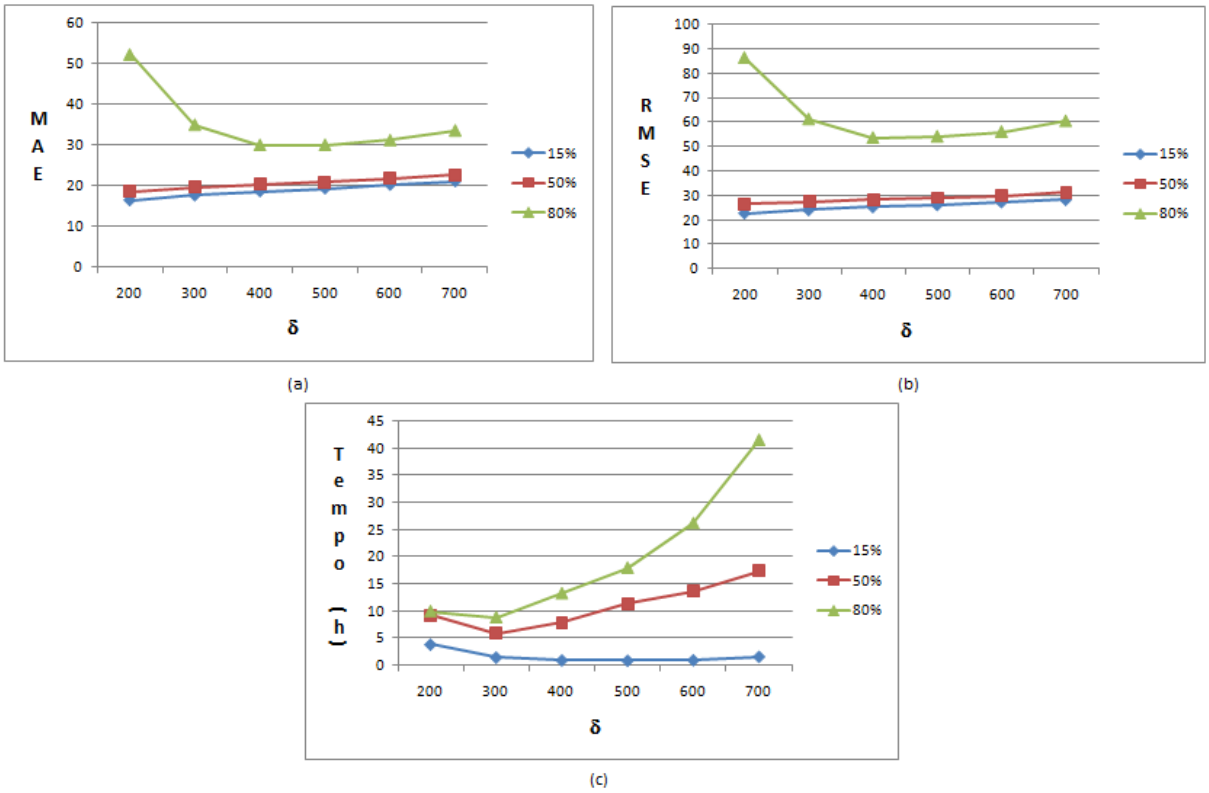


Figura 6.1: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Yeast (MCAR) com a variação de δ.

A Figura 6.2 mostra o comportamento de RMSE e do tempo de execução com a variação de δ. Analisando a base de dados com 15% de dados faltantes, no que diz respeito ao RMSE e ao tempo de execução, percebe-se que δ = 300 foi uma boa opção, pois possui um RMSE aceitável (apenas 5,64% pior que δ = 200) e um tempo de execução menor (1,49 horas). Para a base com 50%, δ = 300 também foi uma boa opção, pois possui o melhor tempo de execução e também um valor de RMSE baixo. Já para 80%, δ = 300 não foi a melhor opção, pois ainda foi possível

melhorar consideravelmente o valor de RMSE. Com δ = 400, conseguiu-se um RMSE e tempo de execução aceitáveis. O valor de RMSE com δ = 400 foi 53,536, o que é 12,49% melhor que com δ = 300, apesar do aumento de 50% no tempo.

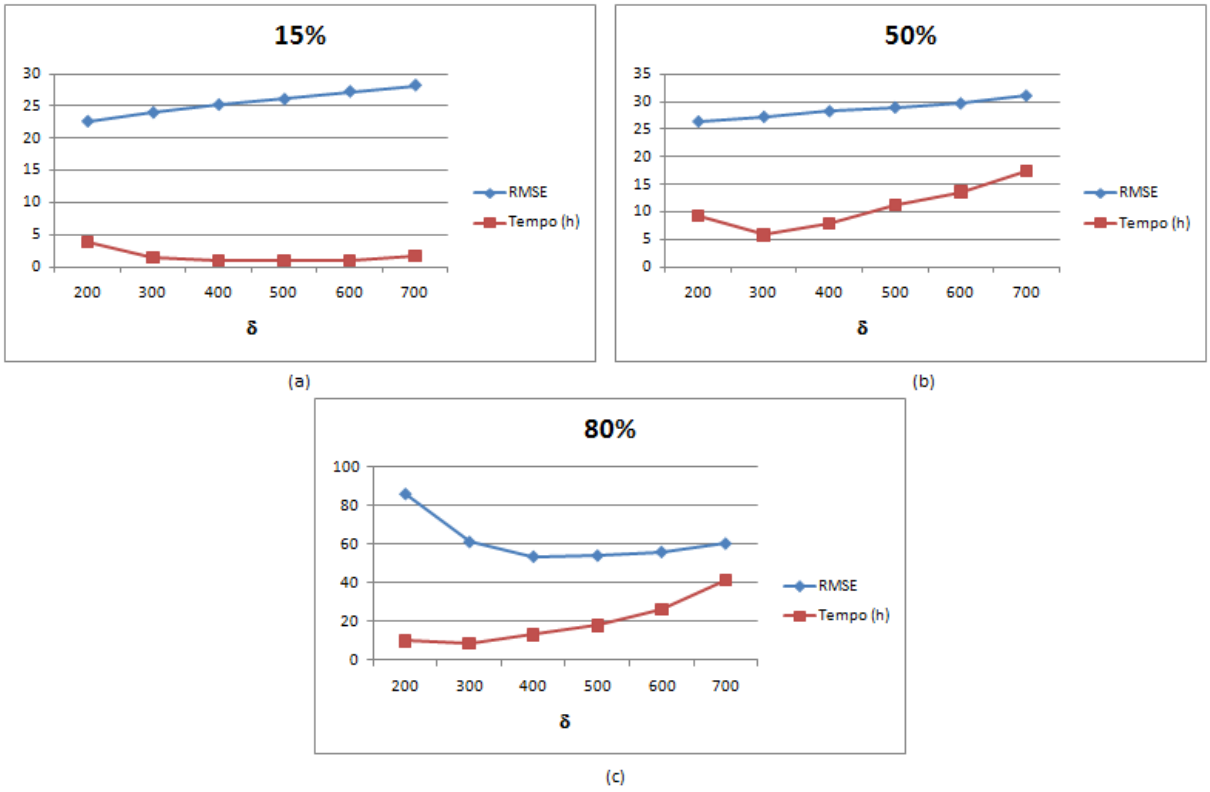


Figura 6.2: Comportamento de RMSE e do tempo de execução com a variação de δ para as bases de dados Yeast (MCAR) com (a) 15%, (b) 50% e (c) 80% de dados faltantes.

6.1.2 Segundo Experimento: a influência de max_it

O segundo experimento investiga a influência da variação do parâmetro max_it no algoritmo SwarmBcluster. Os parâmetros utilizados são os mesmos apresentados na Tabela 6.2, com exceção de max_it, o qual foi variado.

Tabela 6.6: Influência da variação de max_it no valor de MAE para a base Yeast (MCAR).

	15%						50%						80%					
max_it	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
1	17,160	16,046	159,767	12,680	0,00	2,53	19,148	19,388	294,000	13,663	0,00	2,04	44,632	59,856	521,860	24,317	0,00	-27,70
3	17,431	16,201	165,781	12,726	0,00	1,00	19,226	18,555	275,325	14,057	0,00	1,64	38,481	53,551	470,000	21,917	0,00	-10,10
5	17,606	16,332	160,803	12,997	0,00	0,00	19,547	18,945	354,161	14,236	0,00	0,00	34,950	50,211	581,113	20,438	0,00	0,00
7	17,545	16,189	176,389	12,938	0,00	0,35	19,348	18,615	234,999	14,247	0,00	1,02	33,119	47,655	510,871	19,508	0,00	5,24
9	17,850	16,521	182,762	13,245	0,00	-1,39	19,618	19,374	347,790	14,349	0,00	-0,36	33,154	48,337	530,117	19,590	0,00	5,14
11	17,773	16,456	161,887	13,064	0,00	-0,95	19,704	19,307	296,653	14,411	0,00	-0,81	31,603	46,211	476,816	18,709	0,00	9,58
13	17,951	16,533	163,716	13,230	0,00	-1,96	19,689	19,238	300,593	14,458	0,00	-0,73	32,863	47,311	566,078	19,359	0,00	5,97
15	17,891	16,609	163,854	13,235	0,00	-1,62	19,717	19,146	325,248	14,447	0,00	-0,87	32,496	47,020	580,347	19,135	0,00	7,02

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com max_it = 5.

A influência da variação de max_it no valor de MAE é mostrada na Tabela 6.6. Para as bases de dados com 15% e 50% de dados faltantes, a variação no valor de max_it influenciou muito pouco o valor de MAE, assim como o valor do desvio-padrão, da mediana e do erro absoluto mínimo. Para a base com 15%, a maior variação de MAE aconteceu com max_it = 1, com uma melhora de 2,53% se comparado a max_it = 5. Para a base com 50%, a maior variação de MAE aconteceu com max_it = 1, com uma melhora de 2,04% se comparado a max_it = 5. Já para a base com 80% de dados faltantes, o valor de MAE foi consideravelmente influenciado pela variação de max_it. Com max_it = 1 e max_it = 15, o valor de MAE foi, respectivamente, 27,70% pior e 7,02% melhor que com max_it = 5. Apesar de max_it = 15 ter apresentado o melhor valor de MAE, apresentou o segundo maior erro absoluto. O melhor valor para o erro absoluto máximo foi obtido com max_it = 3, aproximadamente 18% menor.

Tabela 6.7: Influência da variação de max_it no valor de RMSE para a base Yeast (MCAR).

	15%			50%			80%		
max_it	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
1	23,493	34,069	2,17	27,249	46,357	-0,10	74,665	130,631	-22,05
3	23,798	34,139	0,90	26,719	42,277	1,84	65,943	122,631	-7,79
5	24,015	34,412	0,00	27,221	44,440	0,00	61,177	121,253	0,00
7	23,873	34,560	0,59	26,849	41,705	1,37	58,034	116,655	5,14
9	24,322	35,568	-1,28	27,572	48,302	-1,29	58,614	120,096	4,19
11	24,222	35,214	-0,86	27,587	45,106	-1,34	55,984	116,027	8,49
13	24,405	35,175	-1,62	27,528	45,443	-1,13	57,605	117,191	5,84
15	24,412	35,621	-1,65	27,484	44,936	-0,96	57,157	117,149	6,57

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com max_it = 5.

A influência da variação de max_it no valor de RMSE é mostrada na Tabela 6.7. Pode-se notar que, para as bases de dados com 15% e 50% de dados faltantes, a variação no valor de max_it influenciou muito pouco o valor de RMSE, assim como o valor do desvio-padrão. Para a base com 15%, a máxima variação do RMSE aconteceu com max_it = 1, com uma melhora de 2,17% se comparado a max_it = 5. Para a base com 50%, a máxima variação de RMSE aconteceu com max_it = 3, com uma melhora de 1,84% se comparado a max_it = 5. Já para a base com 80% de dados faltantes, o valor de RMSE foi consideravelmente influenciado pela variação de max_it. Com max_it = 1 e max_it = 15, o valor de RMSE foi, respectivamente, 22,05% pior e 6,57% melhor que max_it = 5. Analisando o valor de RMSE e do desvio-padrão conjuntamente,

observa-se que esses valores possuem correlação negativa com o valor de max_it (o coeficiente de correlação é igual a -0,806).

A influência da variação de max_it no tempo de execução é mostrada na Tabela 6.8. Como era esperado, para todas as bases de dados, o aumento no valor de max_it aumentou o tempo de execução do algoritmo.

Tabela 6.8: Influência da variação de max_it no Tempo de Execução para a base Yeast (MCAR).

	15%		50%		80%	
max_it	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**
1	0,43	71,53	1,89	67,63	3,07	64,96
3	1,02	31,46	4,42	24,29	6,01	31,38
5	1,49	0,00	5,84	0,00	8,75	0,00
7	2,15	-43,61	7,29	-24,83	11,03	-26,01
9	2,72	-82,23	8,12	-39,13	12,62	-44,17
11	3,37	-125,53	10,17	-74,26	14,09	-60,94
13	3,66	-145,01	12,38	-112,13	15,90	-81,61
15	3,72	-148,70	12,72	-117,95	18,50	-111,29

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com max_it = 5.

A Figura 6.3 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução, com a variação no valor de max_it. Vale notar que, com relação ao MAE e ao RMSE, os resultados para as bases de dados com 15% e 50% de dados faltantes foram bastante estáveis. Para essas mesmas bases de dados, o tempo de execução foi bastante influenciado pelo valor de max_it. Portanto, max_it = 1 é uma boa opção para essas bases de dados. Já para a base de dados com 80% de dados faltantes, a variação de max_it não influenciou só o tempo de execução, mas também melhorou o valor de MAE e RMSE. Portanto, avaliar o melhor valor de max_it para essa base de dados, que possui alta porcentagem de dados faltantes, é uma tarefa mais difícil. É necessário levar em conta o que é mais importante: a qualidade da imputação ou o custo computacional.

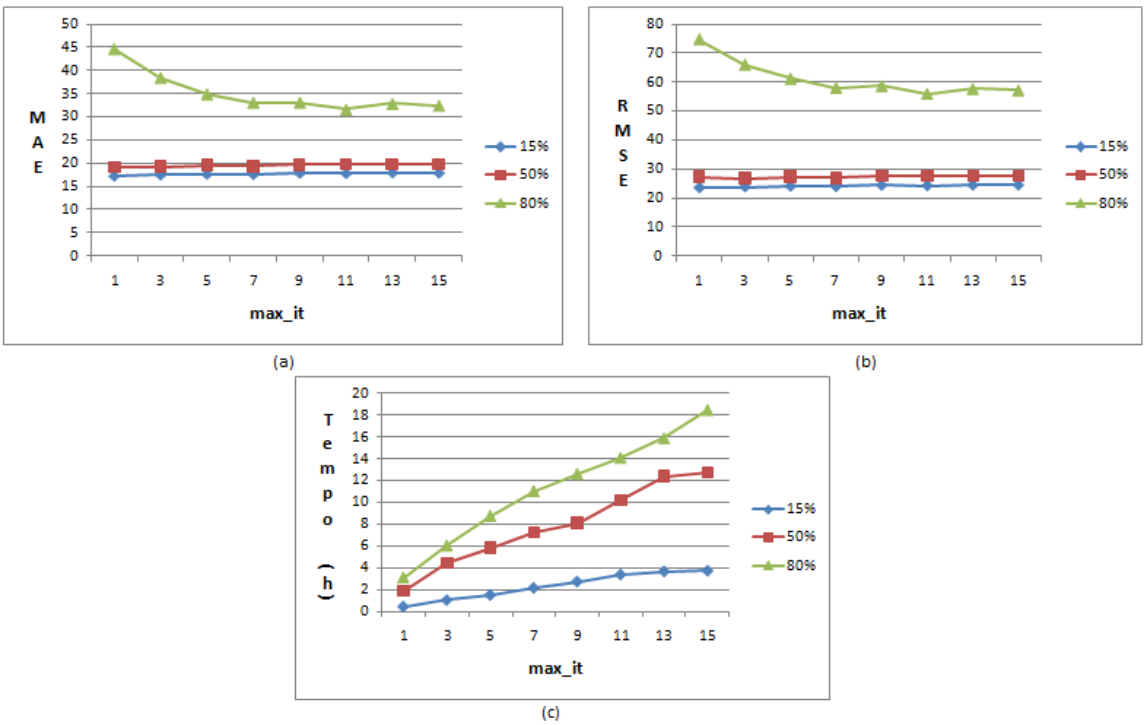


Figura 6.3: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Yeast (MCAR) com a variação de max_it.

6.1.3 Terceiro Experimento: a influência de n_ants

O terceiro experimento investiga a influência da variação do parâmetro n_ants no algoritmo SwarmBcluster. Os parâmetros utilizados são os mesmos apresentados na Tabela 6.2, com exceção de n_ants, o qual foi variado.

A influência da variação de n_ants no valor de MAE é mostrada na Tabela 6.9. A variação no valor de n_ants influenciou pouco o valor de MAE, principalmente para as bases de dados com 15% e 50% de dados faltantes. Para a base de dados com 80% de dados faltantes, a influência só foi significativa para o teste com n_ants = 1, no qual houve uma piora de 9,31% se comparado com n_ants = 5. Para a base com 15%, a maior variação de MAE aconteceu com n_ants = 3, com uma piora de 1,22% quando comparado a n_ants = 5. Para a base com 50%, a maior variação de MAE aconteceu com n_ants = 11, com uma melhora de apenas 1,02% se comparado a n_ants = 5. O erro absoluto máximo não apresentou correlação com o valor de n_ants. Para todos os valores de n_ants, foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Tabela 6.9: Influência da variação de n_ants no valor de MAE para a base Yeast (MCAR).

	15%						50%						80%					
n_ants	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
1	17,655	16,401	167,215	13,016	0,00	-0,27	19,476	19,242	259,634	14,158	0,00	0,36	38,205	52,021	589,297	22,044	0,00	-9,31
3	17,821	16,421	156,490	13,184	0,00	-1,22	19,667	19,877	399,043	14,182	0,00	-0,62	35,012	49,715	539,680	20,438	0,00	-0,18
5	17,606	16,332	160,803	12,997	0,00	0,00	19,547	18,945	354,161	14,236	0,00	0,00	34,950	50,211	581,113	20,438	0,00	0,00
7	17,573	16,307	162,482	13,076	0,00	0,19	19,543	19,925	355,347	14,150	0,00	0,02	36,287	51,086	534,741	21,041	0,00	-3,82
9	17,642	16,471	164,423	13,091	0,00	-0,20	19,491	19,704	360,955	14,057	0,00	0,28	35,162	50,145	487,504	20,392	0,00	-0,60
11	17,504	16,170	164,663	12,932	0,00	0,58	19,347	18,967	240,000	13,971	0,00	1,02	36,421	51,094	489,354	21,204	0,00	-4,21
13	17,666	16,257	174,935	13,106	0,00	-0,34	19,545	19,249	269,459	14,115	0,00	0,01	34,549	48,561	491,000	20,594	0,00	1,15
15	17,626	16,354	153,329	12,989	0,00	-0,11	19,403	19,146	430,865	14,122	0,00	0,73	35,355	49,924	468,641	20,465	0,00	-1,16

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com n_ants = 5.

A influência da variação de n_ants no valor de RMSE é mostrada na Tabela 6.10. A variação no valor de n_ants influenciou pouco o valor de RMSE. Para a base com 15%, a máxima variação de RMSE aconteceu com n_ants = 3, com uma piora de 0,91% se comparado a n_ants = 5. Para a base com 50%, a máxima variação de RMSE aconteceu com n_ants = 7, com uma piora de 2,53% se comparado a n_ants = 5. Para a base com 80%, a máxima variação de RMSE aconteceu com n_ants = 1, com uma piora de 5,50% se comparado a n_ants = 5.

Tabela 6.10: Influência da variação de n_ants no valor de RMSE para a base Yeast (MCAR).

	15%			50%			80%		
n_ants	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
1	24,097	34,825	-0,34	27,379	44,715	-0,58	64,5431	121,79	-5,50
3	24,233	34,819	-0,91	27,962	49,961	-2,72	60,8068	119,10	0,61
5	24,015	34,412	0,00	27,221	44,440	0,00	61,177	121,253	0,00
7	23,973	34,713	0,17	27,909	50,853	-2,53	62,6616	120,59	-2,43
9	24,136	35,020	-0,50	27,716	50,353	-1,82	61,2444	118,96	-0,11
11	23,830	34,500	0,77	27,094	43,600	0,47	62,7466	120,55	-2,57
13	24,008	34,404	0,03	27,433	44,827	-0,78	59,5969	117,11	2,58
15	24,044	34,380	-0,12	27,259	47,489	-0,14	61,1752	118,15	0,00

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com n_ants = 5.

Tabela 6.11: Influência da variação de n_ants no Tempo de Execução para a base Yeast (MCAR).

	15%		50%		80%	
n_ants	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**
1	0,38	74,42	3,07	47,34	5,65	35,47
3	0,95	36,18	4,41	24,36	7,98	8,79
5	1,49	0,00	5,84	0,00	8,75	0,00
7	2,27	-52,10	7,65	-31,10	10,99	-25,53
9	2,67	-78,67	8,48	-45,35	11,14	-27,32
11	3,50	-134,29	10,26	-75,78	13,87	-58,43
13	3,67	-145,54	11,61	-98,87	14,67	-67,61
15	4,83	-223,30	12,76	-118,54	16,68	-90,55

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com n_ants = 5.

A influência da variação de n_ants no tempo de execução é mostrada na Tabela 6.11. Como era esperado, o aumento no valor de n_ants aumentou o tempo de execução.

A Figura 6.4 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução com a variação no valor de n_ants. Vale notar que, com relação ao MAE e ao RMSE, os resultados

para as bases de dados com 15% e 50% de dados faltantes foram bastante estáveis. Esses valores não foram tão estáveis para a base de dados com 80%, mas a variação foi pequena, principalmente no intervalo [3,15]. Como a influência no valor de MAE e RMSE foi muito pequena para as bases de dados com 15% e 50% de dados faltantes, n_ants = 1 representa uma boa opção para essas bases de dados. Para a base de dados com 80% de dados faltantes, n_ants = 3 representa uma boa opção, pois tem um ganho de mais de 6% com relação a n_ants = 1.

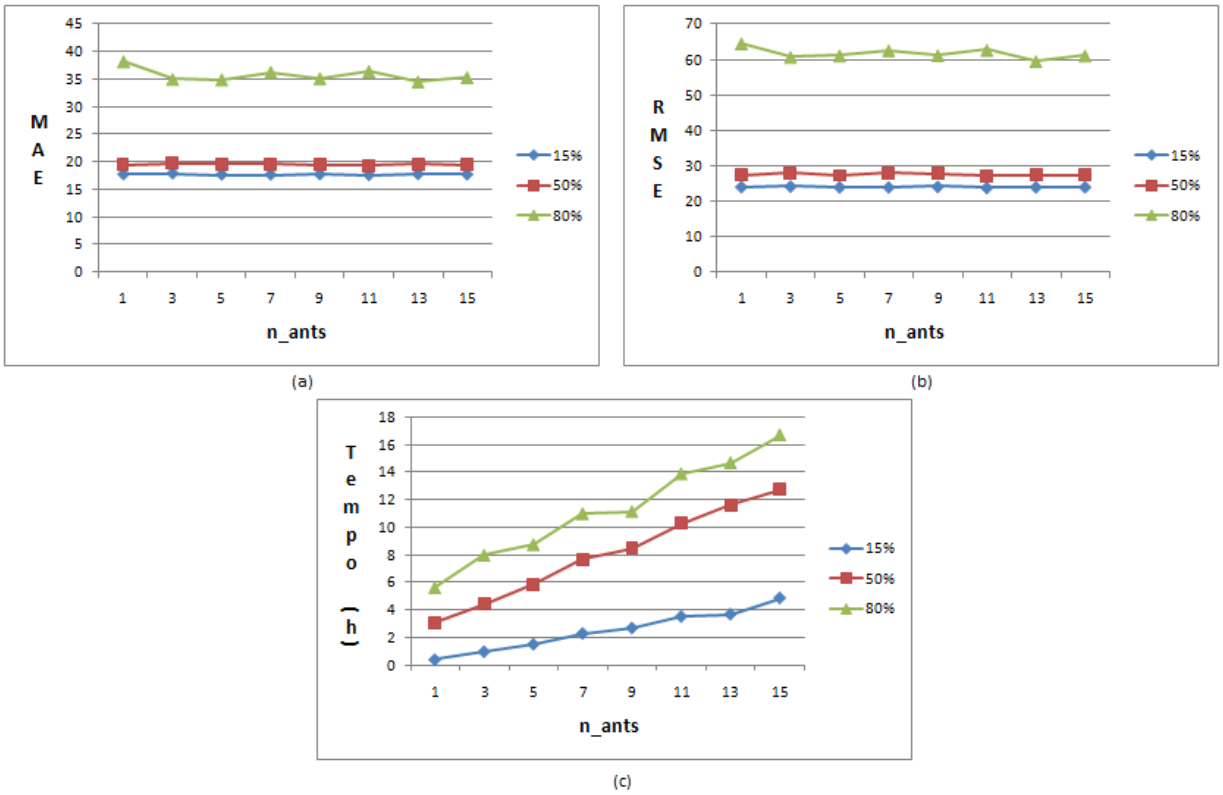


Figura 6.4: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Yeast (MCAR) com a variação de n_ants.

6.1.4 Quarto Experimento: o desempenho da heurística

O quarto experimento investiga o comportamento exclusivamente da heurística construtiva descrita na Seção 5.3. Para isso, foram utilizados max_it = 1 e n_ants = 1. Dessa forma, o algoritmo ACO produzirá biclusters em uma única iteração e utilizando apenas uma única formiga.

Através da análise de sensibilidade paramétrica, observou-se que o parâmetro que exerceu maior influência sobre os valores de MAE e RMSE foi δ. Por isso, a heurística construtiva foi executada utilizando os melhores valores de δ com relação ao RMSE. Logo, para as bases de dados com 15% e 50% de dados faltantes, foi utilizado δ = 200 e para a base com 80% foi utilizado δ = 400. Os resultados são exibidos na Tabela 6.12.

Obviamente, em termos de tempo de execução, os resultados foram consideravelmente melhores que os resultados obtidos na Subseção 6.1.1 (a qual analisou a influência de δ no desempenho do SwarmBcluster e utilizou max_it = 5 e n_ants = 5) e descritos na Tabela 6.5. Em termos de qualidade de imputação, os resultados obtidos com este experimento também foram bons. Se comparados aos testes realizados na Subseção 6.1.1, os resultados aqui obtidos para as bases de dados com 15% e 50% de dados faltantes foram muito próximos, com a vantagem de ter um custo computacional consideravelmente menor. Para a base de dados com 80% de dados faltantes, o valor de MAE obtido neste teste foi 10,73% pior e o valor de RMSE foi 6,53% pior que os valores reportados na Subseção 6.1.1. Em contrapartida, o tempo de execução foi 57,24% melhor.

Tabela 6.12: Comportamento da heurística construtiva com o melhor δ para a base Yeast (MCAR).

	15%		**50%**		**80%**	
MAE	16,278	16,416	18,392	18,525	33,171	29,956
D. Padrão	15,839	15,620	19,206	18,779	46,392	44,371
Máximo	162,371	156,874	328,086	270,134	525,360	584,997
Mediana	11,394	11,891	12,611	13,007	19,970	18,408
Mínimo	0,00	0,00	0,00	0,00	0,00	0,00
RMSE	22,712	22,660	26,592	26,378	57,031	53,536
D. Padrão	33,346	33,017	46,875	43,122	114,973	117,323
Tempo (h)	0,18	3,86	0,68	9,24	5,65	13,21

Nota: Os resultados da Subseção 6.1.1 estão reproduzidos em cinza.

Esses resultados mostram que, utilizando um valor de δ adequado, é possível obter boas estimativas com um custo computacional bastante reduzido.

6.1.5 Quinto Experimento: comparação de desempenho com respeito ao mecanismo MCAR

Este experimento visa avaliar o desempenho do SwarmBcluster com respeito ao mecanismo MCAR. Para isso, seu desempenho foi comparado ao KNNImpute e ao rSVD, todos descritos no Capítulo 5.

A primeira etapa desse experimento é bastante similar com o experimento realizado em DE FRANÇA (2010). A única diferença é que a versão do SwarmBcluster utilizada em DE FRANÇA (2010) não levava em consideração os dados faltantes não-cobertos por biclusters no cálculo dos valores de MAE e RMSE, o que ocasionou uma pequena variação nos valores de MAE e RMSE. A versão utilizada neste trabalho considera tais dados faltantes, os quais são estimados através do vizinho mais próximo, como explicado na Subseção 5.3.1.

O desempenho dos três algoritmos foi avaliado em testes realizados com doze conjuntos de dados faltantes produzidos artificialmente, segundo o mecanismo MCAR, com taxas de 2%, 5%, 10%, 15%, 20%, 30%, 40%, 50%, 60%, 70%, 80% e 90%. Na primeira etapa desse experimento, os parâmetros utilizados foram os apresentados na Tabela 6.2. Em seguida, para as bases de 15%, 50% e 80% de dados faltantes, as quais foram utilizadas nos experimentos de sensibilidade paramétrica, será apresentado o ganho de desempenho do SwarmBcluster com relação ao KNNImpute e ao rSVD.

A Tabela 6.13 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. O algoritmo KNNImpute não foi capaz de tratar as bases de dados com mais de 50% dos dados faltantes. Observa-se que o SwarmBcluster foi superior aos outros algoritmos, exceto para o caso com 90% de dados faltantes. Nesse caso, o rSVD foi 4,18% melhor, mas assim como o SwarmBcluster, o rSVD obteve um resultado bastante ruim. Para os outros casos, o SwarmBcluster obteve desempenho até 61,29% melhor que o KNNImpute e 37,20% melhor que o rSVD. Analisando o valor de MAE juntamente com o valor do desvio-padrão, o SwarmBcluster também foi inferior ao rSVD apenas para o caso de 90%. Para o valor máximo do erro absoluto, o SwarmBcluster foi superior ao KNNImpute em quase todos os casos, mas obteve um resultado equilibrado com relação ao rSVD. Os resultados para a mediana do erro absoluto se comportaram da mesma maneira que os resultados para o valor de MAE. Em todos os testes realizados nesse experimento, os três algoritmos conseguiram estimar valores idênticos, ou

Tabela 6.13: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Yeast (MCAR).

	SwarmBcluster					**KNNImpute**					**rSVD**					**%Melhora**	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
2	18,720	17,464	123,398	12,962	0,00	28,300	25,121	253,929	21,984	0,00	20,625	17,983	113,011	15,749	0,00	51,17	10,18
5	18,512	16,946	131,574	13,765	0,00	28,615	25,195	203,770	22,068	0,02	22,127	19,777	168,448	17,180	0,00	54,57	19,53
10	17,899	16,723	202,693	13,236	0,00	28,671	25,134	265,296	22,268	0,01	22,548	20,344	187,112	17,095	0,00	60,18	25,97
15	17,606	16,332	160,803	12,997	0,00	28,397	24,336	231,298	22,122	0,01	22,330	19,813	188,245	17,071	0,00	61,29	26,83
20	17,879	16,475	203,665	13,215	0,00	28,363	24,990	206,510	21,960	0,00	23,011	20,663	194,379	17,631	0,00	58,64	28,71
30	18,031	16,829	298,125	13,460	0,00	28,674	25,038	315,824	22,578	0,00	23,667	21,428	322,858	18,259	0,00	59,03	31,25
40	18,172	17,128	291,331	13,426	0,00	28,666	25,328	314,769	22,215	0,00	24,933	23,337	322,634	18,745	0,00	57,75	37,20
50	19,547	18,945	354,161	14,236	0,00	28,864	25,411	243,439	22,369	0,00	26,770	24,489	265,306	20,394	0,00	47,67	36,96
60	22,076	24,958	517,192	15,557	0,00						28,371	26,422	302,321	21,510	0,00		28,51
70	26,846	33,467	513,464	17,752	0,00						30,524	29,963	348,214	22,555	0,00		13,70
80	34,950	50,211	581,113	20,438	0,00						37,550	47,733	530,831	24,750	0,00		7,44
90	70,776	89,738	535,024	35,670	0,00						67,814	90,620	581,080	34,027	0,00		-4,18

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

muito próximos, aos valores reais da base de dados, como é possível observar através do valor mínimo do erro absoluto.

A Tabela 6.14 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Com relação ao valor de RMSE, o SwarmBcluster foi superior ao KNNImpute e foi levemente inferior ao rSVD apenas para os casos com 70% ou mais de dados faltantes. O SwarmBcluster foi até 55,73% melhor que o KNNImpute e, para os casos com no máximo 60% de dados faltantes, foi até 36,76% melhor que o rSVD. Analisando o valor de RMSE juntamente com o valor do desvio-padrão, observa-se que o SwarmBcluster foi levemente inferior ao rSVD apenas para os casos com 60% ou mais de dados faltantes.

Tabela 6.14: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Yeast (MCAR).

	SwarmBcluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
2	25,602	35,680	37,841	55,713	27,364	35,910	47,81	6,88
5	25,097	34,786	38,127	53,103	29,677	42,308	51,92	18,25
10	24,496	37,397	38,128	53,653	30,369	43,648	55,65	23,98
15	24,015	34,412	37,398	50,668	29,853	41,345	55,73	24,31
20	24,312	34,776	37,802	51,838	30,927	43,985	55,49	27,21
30	24,664	38,801	38,067	54,026	31,926	47,295	54,34	29,44
40	24,972	39,059	38,253	54,431	34,151	51,328	53,18	36,76
50	27,221	44,440	38,456	53,596	36,282	53,173	41,28	33,29
60	33,321	70,466			38,769	58,122		16,35
70	42,903	86,430			42,773	68,202		-0,30
80	61,177	121,253			60,732	119,270		-0,73
90	114,290	175,208			113,185	180,766		-0,97

Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.15 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base de dados Yeast (MCAR). O SwarmBcluster tem um custo computacional maior que os outros algoritmos. A razão para este alto custo é que o SwarmBcluster cria vários modelos locais da base de dados para encontrar o subconjunto de objetos e atributos mais adequados para imputar um subconjunto de dados faltantes. Já os algoritmos KNNImpute e rSVD criam apenas um único modelo global para toda a base de dados. Por outro lado, essa também pode ser a razão para, na maioria das vezes, o SwarmBcluster obter

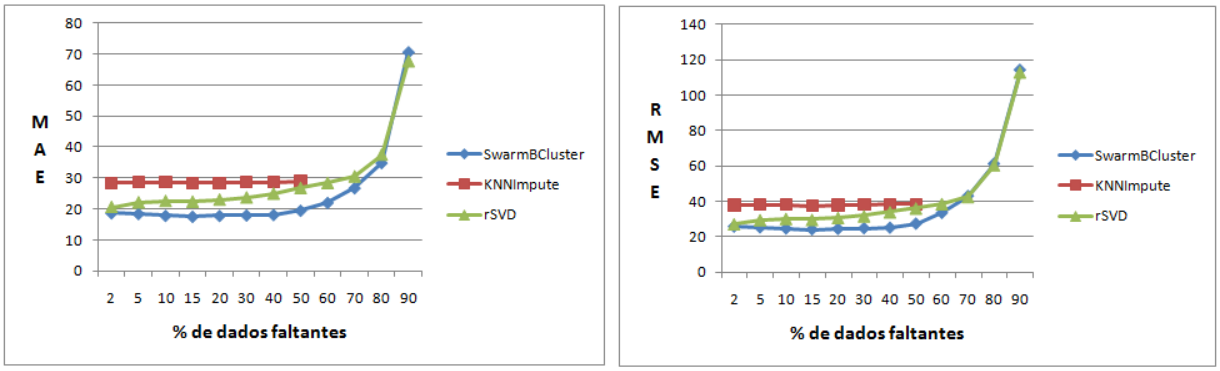
melhor qualidade nas imputações realizadas, uma vez que os modelos locais sofrem menos influência do ruído do conjunto de dados e tendem a selecionar os objetos e atributos mais informativos para a imputação dos dados faltantes (DE FRANÇA, 2010).

Tabela 6.15: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao Tempo de Execução para a base Yeast (MCAR).

	SwarmBcluster		**KNNImpute**	**rSVD**
%	**Tempo (s)**	**Tempo (h)**	**Tempo (s)**	**Tempo (s)**
2	850	0,24	12,09	46,94
5	1.887	0,52	21,84	46,78
10	4.510	1,25	35,88	47,49
15	5.357	1,49	48,50	46,70
20	9.171	2,55	61,17	42,84
30	13.584	3,77	83,89	42,13
40	17.593	4,89	101,28	56,38
50	21.011	5,84	115,02	18,44
60	24.885	6,91		23,00
70	32.026	8,90		16,19
80	31.513	8,75		14,34
90	23.646	6,57		4,67

A Figura 6.5 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Através dela, é possível verificar que os comportamentos de MAE e RMSE são bastante parecidos. Ademais, observa-se que o KNNImpute foi bastante estável para os casos em que ele conseguiu tratar os dados faltantes. O rSVD apresentou um crescimento pequeno de MAE e RMSE até 40% de dados faltantes, a taxa de crescimento é um pouco maior até 80% de dados faltantes e aumenta significativamente com 90%. O SwarmBcluster foi bastante estável até 50% de dados faltantes, os resultados começam a piorar com 60% e há uma piora acentuada com 90%.

A Figura 6.6 mostra o comportamento do valor de MAE, RMSE e do tempo de execução para o algoritmo SwarmBcluster. O tempo de execução apresentou um comportamento diferente do comportamento exibido pelo MAE e RMSE. Ele foi levemente crescente até 70% de dados faltantes, depois disso o tempo de execução começou a diminuir.



(a) (b)

Figura 6.5: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD quando aplicados à base de dados Yeast (MCAR).

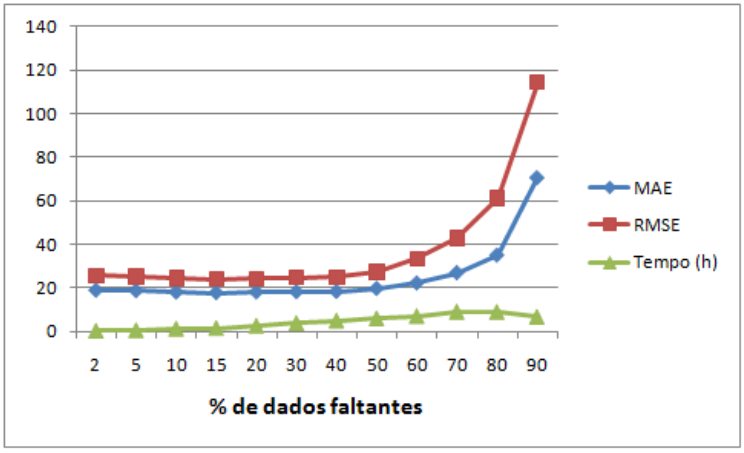


Figura 6.6: Comportamento do valor de MAE, RMSE e do Tempo de Execução para o algoritmo SwarmBcluster quando aplicado à base de dados Yeast (MCAR).

A Figura 6.7 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD para as bases de dados com 15%, 50% e 80% de dados faltantes. Em cada caixa, a marca central é a mediana, as extremidades da caixa são os percentis de 25% e 75%. O bigode se estende aos pontos mais extremos que não são considerados *outliers*. Os outliers são representados individualmente. Para todas essas bases de dados, o SwarmBcluster apresentou erros absolutos mais concentrados, com menor variação.

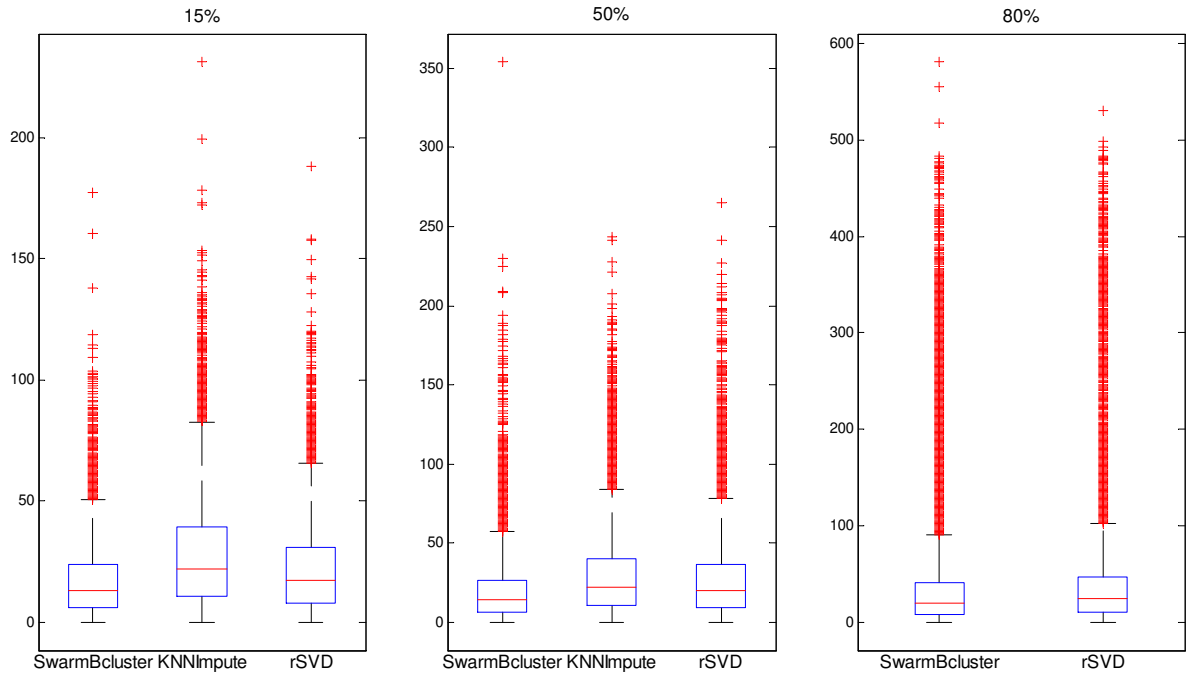


Figura 6.7: Erros absolutos produzidos por SwarmBcluster, KNNImpute e rSVD para a base de dados Yeast (MCAR) com 15%, 50% e 80% de dados faltantes.

Através da análise de sensibilidade paramétrica, observou-se que a variação de δ foi o experimento que trouxe maiores ganhos em termos de MAE e RMSE, ou seja, em termos de qualidade de imputação. Por isso, a comparação de desempenho entre o SwarmBcluster e o KNNImpute e o rSVD é exibida novamente para as bases de dados que participaram da análise de sensibilidade paramétrica. Dessa vez, os melhores valores encontrados para δ foram utilizados. Logo, para a base de dados com 15% e 50% de dados faltantes foi utilizado δ = 200 e para a base de dados com 80% de dados faltantes foi utilizado δ = 400. Os outros parâmetros são iguais àqueles apresentados na Tabela 6.2.

A Tabela 6.16 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. Para a base de dados com 15% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 61,29% para 72,98% e com relação ao rSVD houve um aumento de 26,83% para 36,03%. Além disso, os valores do desvio-padrão, do máximo erro absoluto e da mediana são menores neste teste do que os obtidos com δ = 300. Para a base de dados com 50% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 47,67% para 55,81% e com relação ao rSVD aumentou de 36,96% para 44,51%. Além disso, os valores do desvio-padrão, do máximo erro absoluto e da mediana

são menores neste teste do que os obtidos com $\delta = 300$. Para a base de dados com 80% de dados faltantes, o ganho no valor de MAE com relação ao rSVD aumentou de 7,44% para 25,35%. Além disso, os valores do desvio-padrão e da mediana são menores neste teste do que os obtidos com $\delta = 300$. O valor máximo do erro absoluto é praticamente igual nos dois testes.

A Tabela 6.17 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Para a base de dados com 15% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 55,73% para 65,04% e com relação ao rSVD aumentou de 24,31% para 31,74%. Além disso, o valor do desvio-padrão é menor neste teste do que o obtido com $\delta = 300$. Para a base de dados com 50% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 41,28% para 45,79% e com relação ao rSVD aumentou de 33,29% para 37,54%. Novamente, o valor do desvio-padrão é menor neste teste do que o obtido com $\delta = 300$. Para a base de dados com 80% de dados faltantes, o ganho no valor de RMSE com relação ao rSVD aumentou de -0,73% para 13,44%. E, assim como nos outros casos, o valor do desvio-padrão é menor neste teste do que o obtido com $\delta = 300$.

Tabela 6.16: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para as bases Yeast (MCAR) utilizadas nos experimentos de sensibilidade paramétrica.

	SwarmBcluster					KNNImpute					rSVD					%MelAnter		%MelAtual	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**	**1 vs. 2**	**1 vs. 3**
15	16,416	15,620	156,874	11,891	0,00	28,397	24,336	231,298	22,122	0,01	22,330	19,813	188,245	17,071	0,00	61,29	26,83	72,98	36,03
50	18,525	18,779	270,134	13,007	0,00	28,864	25,411	243,439	22,369	0,00	26,770	24,489	265,306	20,394	0,00	47,67	36,96	55,81	44,51
80	29,956	44,371	584,997	18,408	0,00						37,550	47,733	530,831	24,750	0,00		7,44		25,35

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto, %MelAnter → porcentagem de melhora anterior e %MelAtual → porcentagem de melhora atual. A porcentagem de melhora indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

Tabela 6.17: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para as bases Yeast (MCAR) utilizadas nos experimentos de sensibilidade paramétrica.

	SwarmBcluster		KNNImpute		rSVD		%MelAnter		%MelAtual	
%	RMSE	D.Pad	RMSE	D.Pad	RMSE	D.Pad	1 vs. 2	1 vs. 3	1 vs. 2	1 vs. 3
15	22,660	33,017	37,398	50,668	29,853	41,345	55,73	24,31	65,04	31,74
50	26,378	43,122	38,456	53,596	36,282	53,173	41,28	33,29	45,79	37,54
80	53,536	117,323			60,732	119,270		-0,73		13,44

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, %MelAnter → porcentagem de melhora anterior e %MelAtual → % melhora de atual. A porcentagem de melhora indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

6.1.6 Sexto Experimento: comparação de desempenho com respeito ao mecanismo MAR

O sexto experimento visa comparar o desempenho do SwarmBcluster com relação ao mecanismo MAR. Para isso, seus resultados foram comparados aos resultados obtidos pelo KNNImpute e pelo rSVD, descritos no Capítulo 5.

Para realizar os testes, foram produzidos quatro base de dados artificiais, a partir da base Yeast, da seguinte maneira:

1. O atributo 2 foi escolhido aleatoriamente para ser o motivo da ausência dos dados;
2. Os atributos 3, 5, 11 e 16 foram escolhidos aleatoriamente para apresentar dados faltantes associados a uma dada probabilidade, sempre que o valor do atributo 2 fosse maior que sua média; e
3. As probabilidades escolhidas foram: 20%, 40%, 60% e 80%, gerando, portanto, os quatro casos artificiais de dados faltantes.

Os parâmetros utilizados para o SwarmBcluster foram praticamente os mesmos dos apresentados na Tabela 6.2, com exceção de δ, cujo valor utilizado foi 200. Esse valor foi escolhido através de alguns testes preliminares que foram realizados.

A Tabela 6.18 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. O SwarmBcluster foi superior aos outros dois algoritmos. Ele foi até 71,42% melhor que o KNNImpute e 29,50% melhor que o rSVD. O mesmo aconteceu para o desvio-padrão e a mediana, ou seja, o SwarmBcluster foi superior aos outros dois algoritmos. Com relação ao erro absoluto máximo, o SwarmBcluster foi melhor que o rSVD e foi pior que o

Tabela 6.18: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Yeast (MAR).

	SwarmBcluster					KNNImpute					rSVD					% Melhora	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
20	17,314	15,968	130,521	12,720	0,00	29,666	26,581	190,366	22,322	0,03	22,423	19,955	182,529	17,222	0,00	71,34	29,50
40	17,428	16,430	142,374	12,939	0,00	29,875	27,124	266,357	22,639	0,00	22,015	19,383	204,654	17,291	0,00	71,42	26,32
60	17,844	16,704	179,784	12,955	0,00	29,360	25,736	258,463	22,575	0,00	22,058	19,431	178,414	17,023	0,00	64,54	23,62
80	19,311	18,686	266,029	13,880	0,00	29,832	26,662	226,577	22,723	0,02	23,070	20,361	198,096	18,094	0,00	54,48	19,47

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

KNNImpute apenas para o caso com 80% de dados faltantes. Em todos os testes realizados nesse experimento, os três algoritmos conseguiram estimar valores idênticos, ou muito próximos, aos valores reais da base de dados, como se pode observar através do valor mínimo do erro absoluto.

A Tabela 6.19 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Novamente, o SwarmBcluster foi superior aos demais algoritmos. Ele foi até 69,12% melhor que o KNNImpute e 27,44% melhor que o rSVD.

Tabela 6.19: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Yeast (MAR).

	SwarmBcluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
20	23,553	33,816	39,832	55,651	30,017	43,654	69,12	27,44
40	23,952	34,535	40,351	57,699	29,332	42,333	68,47	22,46
60	24,443	35,321	39,043	54,287	29,396	41,231	59,73	20,26
80	26,872	41,881	40,010	56,011	30,770	43,740	48,89	14,51

Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.20 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução. Como era esperado, o SwarmBcluster tem um custo computacional maior que os outros algoritmos.

Tabela 6.20: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base Yeast (MAR).

	SwarmBcluster		**KNNImpute**	**rSVD**
%	**Tempo (s)**	**Tempo (h)**	**Tempo (s)**	**Tempo (s)**
20	4267,94	1,19	19,13	46,16
40	8917,36	2,48	32,31	45,13
60	11968,17	3,32	41,80	41,74
80	12577,74	3,49	49,05	49,08

A Figura 6.8 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Através dela, é possível verificar que os comportamentos de MAE e RMSE são bastante parecidos. O desempenho dos três algoritmos foi estável, mas o SwarmBcluster foi visivelmente melhor que os outros dois algoritmos.

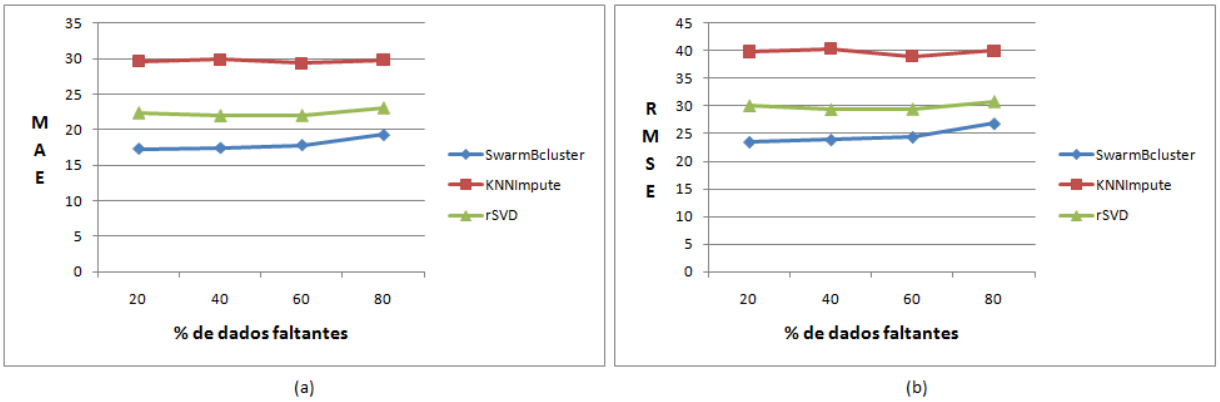


Figura 6.8: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD para a base de dados Yeast (MAR).

A Figura 6.9 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD, para as bases de dados com 40% e 80% de dados faltantes. É possível visualizar que o SwarmBcluster apresentou erros absolutos com menor variação.

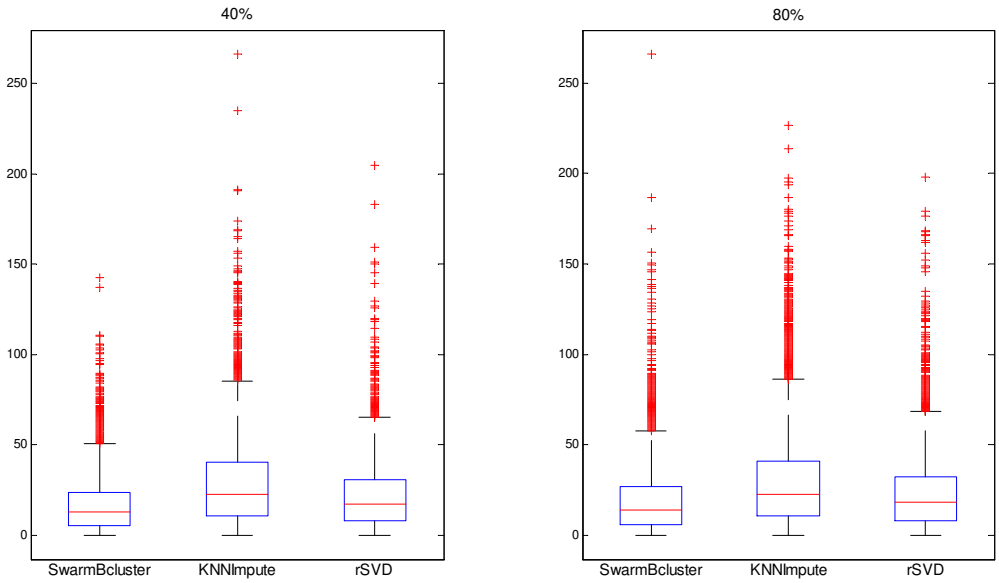


Figura 6.9: Erros absolutos produzidos por SwarmBcluster, KNNImpute e rSVD para a base de dados Yeast (MAR) com 40% e 80% de dados faltantes.

6.1.7 Sétimo Experimento: comparação de desempenho com respeito ao mecanismo MNAR

O sétimo experimento visa comparar o desempenho do SwarmBcluster com relação ao mecanismo MNAR. Para isso, seus resultados foram comparados aos resultados obtidos pelo KNNImpute e rSVD, descritos no Capítulo 5.

Para realizar os testes, foram produzidos quatro bases de dados artificiais, da seguinte maneira:

1. O atributo 7 foi escolhido aleatoriamente para ser o motivo da ausência dos dados. Esse atributo tem certa probabilidade de possuir dados faltantes quando o seu valor é maior que sua média.
2. As probabilidades escolhidas foram: 20%, 40%, 60% e 80%, gerando, portanto, as cinco bases de dados artificiais.

Os parâmetros utilizados para o SwarmBcluster são os mesmos apresentados na Tabela 6.2.

A Tabela 6.21 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. O SwarmBcluster produziu os melhores valores para MAE. Ele foi até 47,44% melhor que o KNNImpute e até 27,01% melhor que o rSVD. Considerando o valor de MAE juntamente com o desvio-padrão, o SwarmBcluster é levemente pior que o rSVD apenas para o caso de 40%. O SwarmBcluster também apresentou valores máximo de erro absoluto melhores que o KNNImpute, mas foi melhor que o rSVD apenas para o caso de 60%. O SwarmBcluster também apresentou valores melhores para a mediana do erro absoluto. Todos os algoritmos apresentaram bons valores para o erro absoluto mínimo.

A Tabela 6.22 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. O SwarmBcluster foi superior aos outros algoritmos, exceto para o caso de 40%, no qual ele foi 5,15% pior que o rSVD. Nesse caso, o SwarmBcluster produziu alguns erros absolutos maiores que os produzidos pelo rSVD. Isso explica porque o valor de MAE produzido pelo SwarmBcluster foi apenas 0,67% melhor que o valor produzido pelo rSVD. Essa melhoria de 0,67% foi bem inferior aos resultados obtidos com os outros conjuntos de dados testados. Para esses, o valor de RMSE produzido pelo SwarmBcluster foi até 43,23% melhor que o KNNImpute e 23,52% melhor que o rSVD. Considerando o valor de RMSE juntamente com o desvio-padrão, o SwarmBcluster ganha do rSVD apenas no caso de 60%.

Tabela 6.21: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Yeast (MNAR).

	SwarmBcluster					KNNImpute					rSVD					% Melhora	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
20	16,675	15,154	134,861	13,322	0,05	24,585	20,908	135,669	19,279	0,07	17,951	14,238	74,827	14,298	0,02	47,44	7,65
40	17,217	15,900	134,790	13,525	0,02	24,908	21,546	156,931	19,760	0,03	17,333	13,918	113,196	14,149	0,00	44,67	0,67
60	17,526	15,476	140,121	13,865	0,02	25,296	20,940	158,709	21,102	0,01	22,259	18,402	143,874	17,832	0,00	44,33	27,01
80	18,535	17,024	136,722	14,215	0,00	26,720	21,309	166,174	22,558	0,02	19,736	15,836	118,090	16,385	0,00	44,16	6,48

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

Tabela 6.22: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Yeast (MNAR).

	SwarmBcluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
20	22,532	33,345	32,273	43,293	22,912	28,498	43,23	1,68
40	23,436	35,772	32,934	45,512	22,229	28,618	40,53	-5,15
60	23,381	32,952	32,839	43,307	28,881	37,858	40,45	23,52
80	25,167	35,900	34,177	44,953	25,304	32,377	35,80	0,54

Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.23 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução. Como era esperado, o SwarmBcluster tem um custo computacional maior que os outros algoritmos. Entretanto, o tempo de execução do SwarmBcluster não passou de 10 minutos, o que pode ser considerado um bom desempenho para a maioria das aplicações.

Tabela 6.23: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base Yeast (MNAR).

	SwarmBcluster		**KNNImpute**	**rSVD**
%	**Tempo (s)**	**Tempo (min)**	**Tempo (s)**	**Tempo (s)**
20	187,36	3,12	8,27	49,64
40	323,64	5,39	11,47	52,28
60	411,80	6,86	14,74	35,24
80	584,66	9,74	16,44	55,75

A Figura 6.10 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Através dela, é possível verificar que os comportamentos de MAE e RMSE foram bastante parecidos. O rSVD obteve o resultado mais instável, mas em dois dos quatro testes obteve resultados bem parecidos com o SwarmBcluster. O KNNImpute foi estável, mas obteve os piores resultados.

A Figura 6.11 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD para as bases de dados com 40% e 80% de dados faltantes. Para a base de dados com 40% de dados faltantes, os erros absolutos obtidos pelo SwarmBcluster e pelo rSVD foram bastante parecidos, apesar do SwarmBcluster ter produzido mais outliers. Para a base de dados com 80% de dados faltantes, os erros absolutos obtidos pelo

SwarmBcluster foram levemente mais concentrados que os obtidos pelo rSVD, mas novamente o SwarmBcluster produziu mais outliers. Em ambos os casos, o SwarmBcluster foi superior ao KNNImpute.

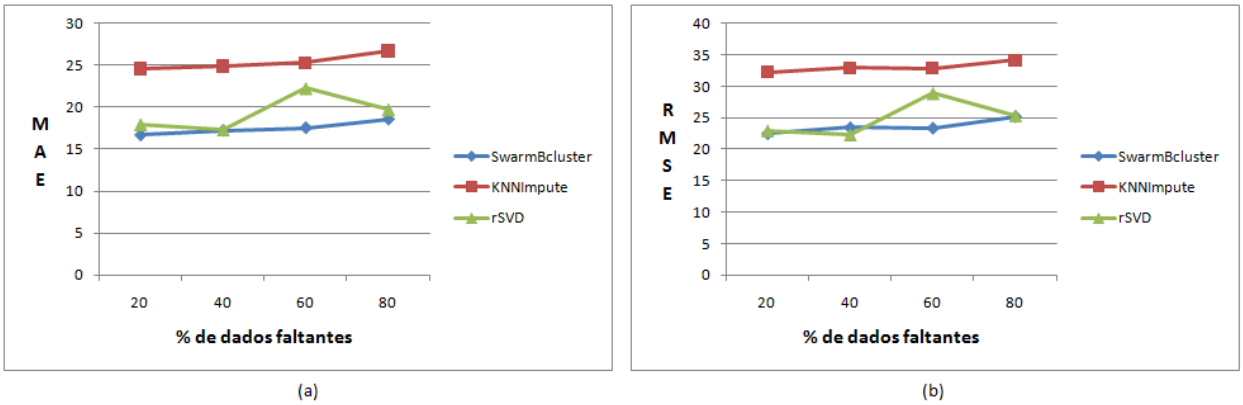


Figura 6.10: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD para a base de dados Yeast (MNAR).

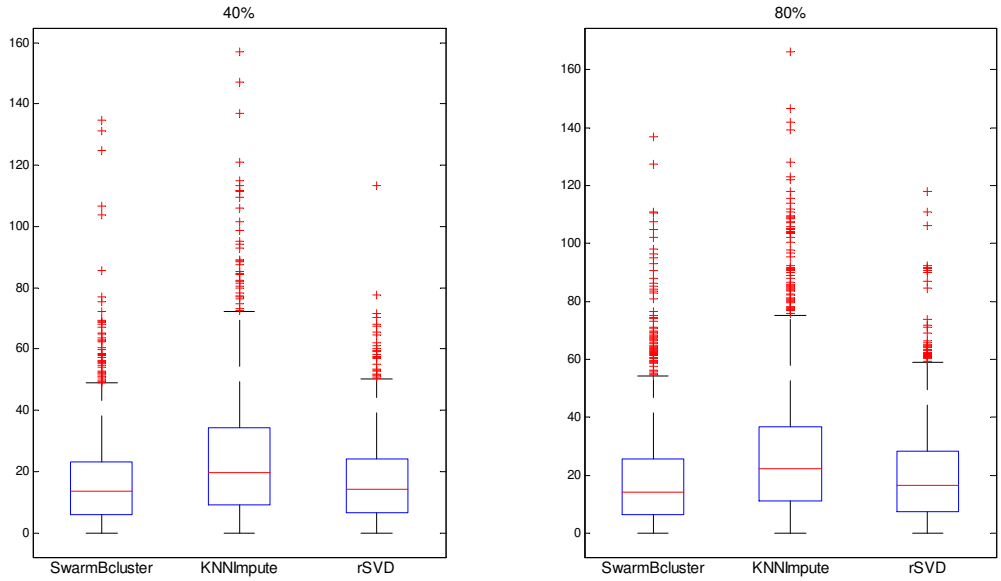


Figura 6.11: Erros absolutos produzidos pelo SwarmBcluster, KNNImpute e rSVD para a base de dados Yeast (MNAR) com 40% e 80% de dados faltantes.

6.2 Base de dados Human

Nesta seção, serão descritos os experimentos realizados com a base de dados Human. Esses experimentos foram realizados em um computador Intel Core^{tm}2 Quad Q6600 @ 2,40 Ghz, 2 GB de RAM, utilizando apenas um único *core* para cada experimento.

Os parâmetros-base para os experimentos que foram realizados são listados a seguir. O parâmetro utilizado para o KNNImpute foi $K = 5$. Como foi mencionado na Seção 5.1, para encontrar um valor de K apropriado para os experimentos deste trabalho, foram realizados alguns experimentos preliminares. Os parâmetros utilizados para o rSVD são os mesmos que foram utilizados nos experimentos com a base de dados Yeast e são apresentados na Tabela 6.1. Os parâmetros utilizados para o SwarmBcluster são apresentados na Tabela 6.24. O valor de $\delta = 1200$ foi determinado por CHENG & CHURCH (2000), baseado nos estudos de clusters realizados nessa base por TAVAZOIE *et al*. (1999). Os valores dos outros parâmetros foram baseados em DE FRANÇA (2010), com exceção de *col_min*. Em DE FRANÇA (2010), o valor utilizado para *col_min* é 20, mas em testes preliminares realizados neste trabalho, verificou-se que esse valor aumentava consideravelmente o custo computacional e não trazia ganhos na qualidade da imputação. Logo, o valor utilizado para *col_min* é 7, o mesmo valor que foi utilizado nos experimentos com a base de dados Yeast.

Tabela 6.24: Parâmetros-base utilizados pelo SwarmBcluster nos experimentos com a base de dados Human.

Parâmetro	**Valor**
δ	1200
max_it	5
n_ants	5
α	1
β	3
ρ	0,2
col_min	7
row_min	100
max_vars	2500

Assim como foi realizado com a base de dados Yeast, a análise de sensibilidade paramétrica, exibida a seguir, foi realizada com bases de dados que possuem 15%, 50% e 80% de dados faltantes segundo o mecanismo MCAR.

6.2.1 Primeiro Experimento: a influência de δ

O primeiro experimento investiga a influência da variação do parâmetro δ no algoritmo SwarmBcluster. Os parâmetros utilizados são os mesmos apresentados na Tabela 6.24, com exceção de δ, o qual foi variado.

A influência da variação de δ no valor de MAE é mostrada na Tabela 6.25.

Para a base com 15% de dados faltantes, a diminuição no valor de δ trouxe resultados melhores para o valor de MAE e do desvio-padrão, enquanto o aumento no valor de δ trouxe resultados piores. Com δ = 600 e δ = 2000, o resultado de MAE foi, respectivamente, 7,31% melhor e 9,86% pior que com δ = 1200. O valor máximo do erro absoluto não apresentou correlação com o valor de δ. A mediana se comportou de forma idêntica ao valor de MAE e do desvio-padrão, isto é, quanto maior o valor de δ, pior o valor da mediana. Para todos os valores de δ, foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Os testes com a base com 50% de dados faltantes apresentaram um comportamento semelhante aos testes com a base de 15%, ou seja, valores menores de δ geraram melhores resultados. Com δ = 600 e δ = 2000, o resultado de MAE foi, respectivamente, 6,73% melhor e 6,57% pior que com δ = 1200. O valor máximo do erro absoluto não apresentou correlação com o valor de δ. O desvio-padrão e a mediana se comportaram de forma idêntica ao valor de MAE. Para todos os valores de δ, foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Tabela 6.25: Influência da variação de δ no valor de MAE para a base Human (MCAR).

	15%						50%						80%					
δ	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
600	30,887	29,386	564,093	23,220	0,00	7,31	33,500	31,969	459,000	24,967	0,00	6,73	42,049	43,571	959,796	31,493	0,00	-1,55
800	31,748	29,860	536,795	24,111	0,00	4,73	34,191	32,707	726,289	25,746	0,00	4,80	41,736	42,495	780,000	31,268	0,00	-0,79
1000	32,660	30,835	564,527	24,593	0,00	1,99	34,855	33,160	621,714	26,068	0,00	2,95	41,757	42,225	797,000	31,267	0,00	-0,84
1200	33,324	31,222	532,361	25,236	0,00	0,00	35,915	34,221	502,147	26,761	0,00	0,00	41,410	41,440	857,000	30,839	0,00	0,00
1400	34,212	32,254	519,816	25,735	0,00	-2,66	36,482	34,676	504,953	27,278	0,00	-1,58	41,732	41,230	748,117	31,032	0,00	-0,78
1600	35,082	32,992	548,761	26,683	0,00	-5,27	37,145	35,334	669,086	27,814	0,00	-3,42	41,963	41,808	808,830	31,079	0,00	-1,34
1800	35,863	33,815	529,525	27,232	0,00	-7,62	37,731	35,904	621,714	28,225	0,00	-5,05	42,226	41,301	628,049	31,272	0,00	-1,97
2000	36,609	34,064	529,550	27,747	0,00	-9,86	38,276	36,345	599,000	28,752	0,00	-6,57	42,770	41,981	823,169	31,757	0,00	-3,28

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com δ = 1200.

Nos testes realizados para a base com 80% de dados faltantes, a variação de δ piorou o valor de MAE. Logo, o melhor valor encontrado foi δ = 1200. Mesmo analisando o valor de δ juntamente com o desvio-padrão, o melhor valor encontrado foi δ = 1200. O valor máximo do erro absoluto não apresentou correlação com o valor de δ. A mediana apresentou o mesmo comportamento que o apresentado pelo valor de MAE. Como nos testes com as bases com 15% e 50% de dados faltantes, para todos os valores de δ foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Tabela 6.26: Influência da variação de δ no valor de RMSE para a base Human (MCAR).

	15%			**50%**			**80%**		
δ	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
600	42,632	71,497	6,64	46,306	72,610	6,66	60,553	112,480	-3,36
800	43,584	69,629	4,56	47,316	77,304	4,62	59,563	106,076	-1,67
1000	44,916	72,893	1,64	48,108	75,678	3,02	59,385	104,488	-1,37
1200	45,665	71,833	0,00	49,608	77,298	0,00	58,583	101,282	0,00
1400	47,019	73,702	-2,97	50,332	77,908	-1,46	58,664	98,147	-0,14
1600	48,158	76,287	-5,46	51,266	80,480	-3,34	59,235	102,368	-1,11
1800	49,291	77,263	-7,94	52,084	82,044	-4,99	59,066	94,806	-0,82
2000	50,006	76,669	-9,51	52,783	81,914	-6,40	59,930	99,121	-2,30

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com δ = 1200.

A influência da variação de δ no valor de RMSE é mostrada na Tabela 6.26.

Para a base com 15% de dados faltantes, a diminuição no valor de δ trouxe resultados melhores para o valor de RMSE, enquanto o aumento no valor de δ trouxe resultados piores. Com δ = 600 e δ = 2000, o resultado de RMSE foi, respectivamente, 6,64% melhor e 9,51% pior que com δ = 1200. O valor do desvio-padrão é crescente para δ ≥ 1200. Para δ < 1200, não existe correlação entre o valor de δ e do desvio-padrão.

Os testes com a base com 50% de dados faltantes apresentaram um comportamento semelhante aos testes com a base com 15%. Com δ = 600 e δ = 2000, o resultado de RMSE foi, respectivamente, 6,66% melhor e 6,4% pior que com δ = 1200. O valor do desvio-padrão foi maior para os maiores valores de δ, exceto para o teste com δ = 1000, cujo desvio-padrão foi menor que o obtido com δ = 800.

Nos testes realizados com a base de dados com 80% de dados faltantes, a variação de δ piorou o valor de RMSE. Logo, o melhor valor encontrado foi δ = 1200. O desvio-padrão apresentou correlação positiva com o valor de δ, de modo que se o RMSE for considerado conjuntamente com o desvio-padrão, o melhor valor de δ passa a ser 1800.

A influência da variação de δ no tempo de execução é mostrada na Tabela 6.27. Para a base com 15% de dados faltantes, o tempo de execução tende a ser maior com valores pequenos de δ. Para a base com 50% de dados faltantes, a variação de δ, para baixo ou para cima, só piorou o tempo de execução. O mesmo aconteceu com o caso de 80%. Apesar disso, nota-se que, para esses dois últimos casos, o aumento do tempo quando δ é diminuído não é tão expressivo quanto para o caso de 15%.

Tabela 6.27: Influência da variação de δ no tempo de execução para a base Human (MCAR).

	15%		**50%**		**80%**	
δ	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**
600	21,01	-130,55	43,05	-111,00	42,44	-54,74
800	13,78	-51,18	27,08	-32,71	30,36	-10,70
1000	8,89	2,42	22,48	-10,17	27,66	-0,85
1200	9,11	0,00	20,40	0,00	27,42	0,00
1400	5,49	39,71	20,84	-2,16	29,31	-6,88
1600	5,05	44,63	20,91	-2,48	29,55	-7,75
1800	4,37	52,00	22,48	-10,17	32,80	-19,62
2000	4,29	52,93	20,68	-1,37	31,60	-15,24

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com δ = 1200.

A Figura 6.12 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução, com a variação no valor de δ. Vale notar que, com relação ao MAE e ao RMSE, os experimentos realizados para os casos com 15% e 50% de dados faltantes possuíram um comportamento semelhante. Ambos, MAE e RMSE, pioraram com o aumento no valor de δ. Já a influência causada por δ nos valores de MAE e RMSE para o caso de 80% foi pequena. Em relação ao custo computacional, para todas as bases de dados, o custo computacional é maior para δ menores e tende a se estabilizar com o crescimento de δ.

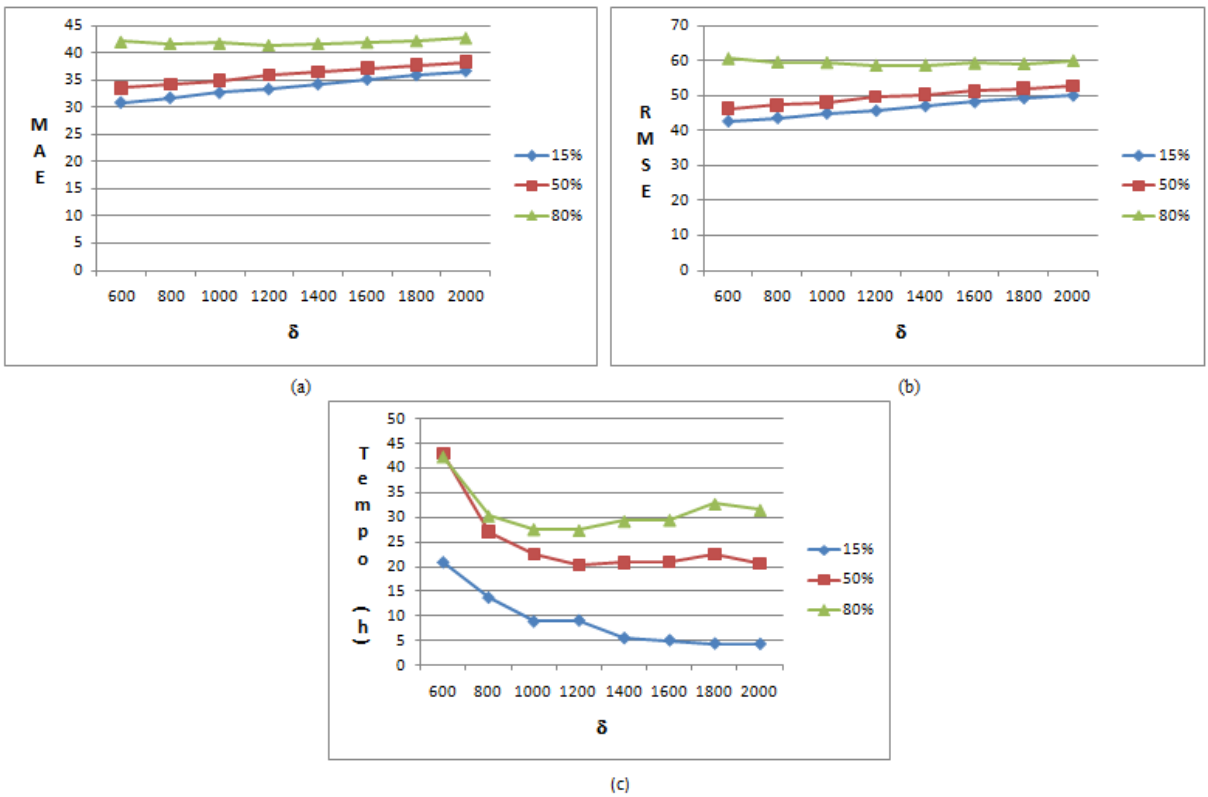


Figura 6.12: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Human (MCAR) com a variação de δ.

A Figura 6.13 mostra o comportamento de RMSE e do tempo de execução com a variação de δ. Os testes realizados com as bases de dados com 15% e 50% de dados faltantes apresentaram RMSE melhor para os menores valores de δ, porém o custo computacional é bem maior para esses valores de δ. O ganho em termos de qualidade da imputação foi bem menor que o aumento no custo computacional. Mas, de qualquer maneira, avaliar o melhor valor de δ para essas bases de dados é uma tarefa difícil. É necessário levar em conta o que é mais importante: a qualidade da imputação ou o custo computacional. Para a base de dados com 80% de dados faltantes, certamente, δ = 1200 foi o melhor valor, pois possui o melhor RMSE e o melhor custo computacional. De qualquer forma, aparentemente existe um compromisso entre o valor de δ e o desempenho do algoritmo, tanto de acuidade como computacional. Esse compromisso é mais acentuado em alguns casos (15%) e menos acentuado em outros (80%).

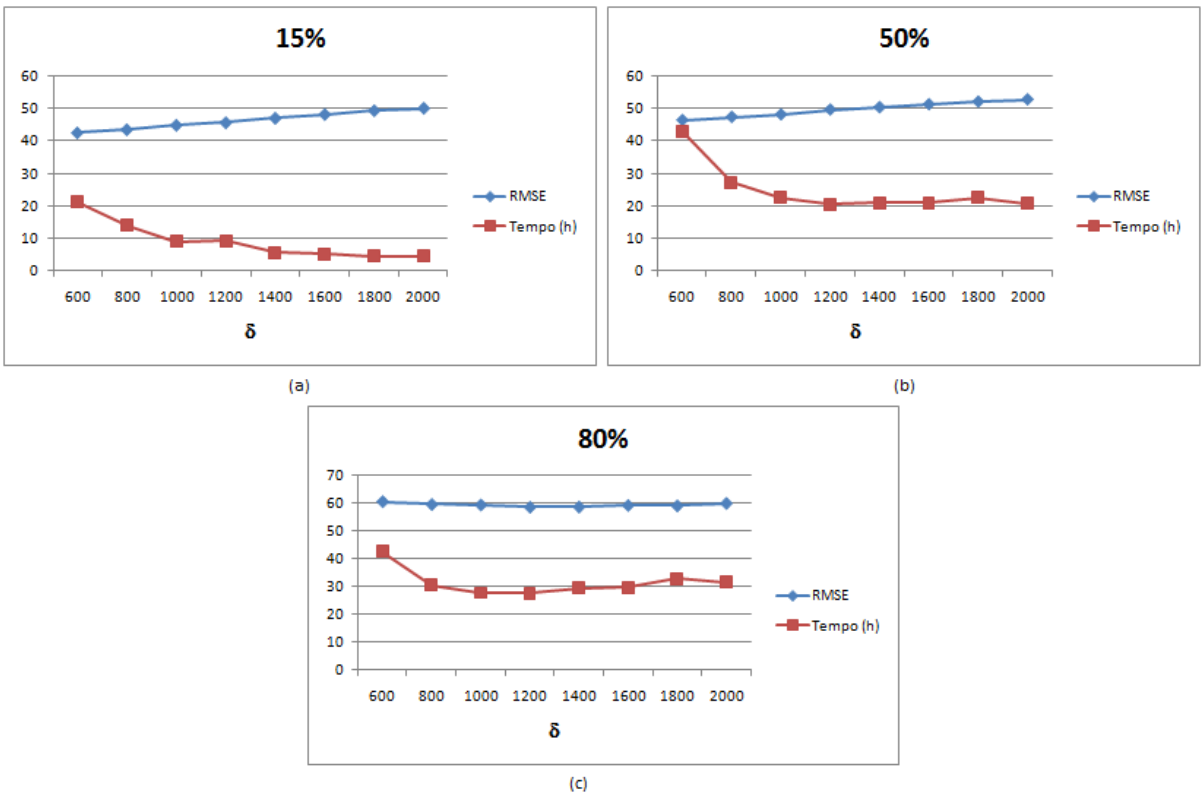


Figura 6.13: Comportamento de RMSE e do tempo de execução com a variação de δ para as bases de dados Human (MCAR) com (a) 15%, (b) 50% e (c) 80% de dados faltantes.

6.2.2 Segundo Experimento: a influência de max_it

O segundo experimento investiga a influência da variação do parâmetro max_it no algoritmo SwarmBcluster. Os parâmetros utilizados são os mesmos apresentados na Tabela 6.24, com exceção de max_it, o qual foi variado.

A influência da variação de max_it no valor de MAE é mostrada na Tabela 6.28. Para todas as bases de dados, a variação desse parâmetro influenciou muito pouco o valor de MAE. Para a base com 15%, a maior variação de MAE aconteceu com max_it = 1, com uma melhora de 1,97% se comparado a max_it = 5. Para a base com 50%, a maior variação de MAE também aconteceu com max_it = 1, com uma melhora de 2,71% se comparado a max_it = 5. Para a base com 80%, a maior variação de MAE também aconteceu com max_it = 1, com uma piora de 1,46% se comparado a max_it = 5. O valor do desvio-padrão e da mediana também sofreram pouca influência do valor escolhido para esse parâmetro. Para as bases de dados com 15% e 50% de dados faltantes, o valor máximo do erro absoluto não apresentou correlação com o valor de max_it. Para a base de dados com 80% de dados faltantes, existe uma pequena correlação

negativa entre max_it e o valor máximo do erro absoluto. Nos testes realizados, para todos os valores de max_it foram obtidas imputações praticamente perfeitas, o que reflete no valor mínimo do erro absoluto.

A influência da variação de max_it no valor de RMSE é mostrada na Tabela 6.29. Pode-se notar que, para todas as bases de dados, a variação no valor de max_it influenciou muito pouco o valor de RMSE. Para a base com 15% de dados faltantes, a máxima variação do RMSE aconteceu com max_it = 1, com uma melhora de 1,8% se comparado a max_it = 5. Para a base com 50%, a máxima variação de RMSE também aconteceu com max_it = 1, com uma melhora de 2,59% se comparado a max_it = 5. Para a base com 80% de dados faltantes, a máxima variação de RMSE também aconteceu com max_it = 1, com uma piora de 1,84% se comparado a max_it = 5. Apenas o desvio-padrão dos testes realizados com a base de dados com 80% de dados faltantes apresentou correlação com o valor de max_it (o coeficiente de correlação é igual a -0,857).

A influência da variação de max_it no tempo de execução é mostrada na Tabela 6.30. Como era esperado, para todas as bases de dados, o aumento no valor de max_it aumentou o tempo de execução do algoritmo.

A Figura 6.14 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução, com a variação no valor de max_it. Vale notar que, com relação ao MAE e ao RMSE, os resultados foram estáveis. Porém, o tempo de execução foi bastante influenciado pelo valor de max_it. Nesse cenário, é perceptível que max_it = 1 é a melhor opção.

Tabela 6.28: Influência da variação de max_it no valor de MAE para a base Human (MCAR).

	15%						50%						80%					
max_it	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
1	32,669	30,720	534,369	24,754	0,00	1,97	34,941	33,378	684,656	26,282	0,00	2,71	42,015	42,356	901,000	31,345	0,00	-1,46
3	33,363	31,282	541,036	25,167	0,00	-0,12	35,584	33,790	513,957	26,632	0,00	0,92	41,710	42,198	764,720	30,924	0,00	-0,73
5	33,324	31,222	532,361	25,236	0,00	0,00	35,915	34,221	502,147	26,761	0,00	0,00	41,410	41,440	857,000	30,839	0,00	0,00
7	33,618	31,559	528,514	25,506	0,00	-0,88	36,069	34,678	786,114	27,0751	0,00	-0,43	41,506	41,244	799,640	30,959	0,00	-0,23
9	33,682	31,590	522,174	25,690	0,01	-1,07	36,068	34,252	509,426	26,923	0,00	-0,43	41,884	42,135	791,676	31,139	0,00	-1,15
11	33,754	31,793	515,202	25,624	0,00	-1,29	36,335	34,593	493,012	27,309	0,00	-1,17	41,492	40,869	711,171	30,865	0,00	-0,20
13	33,901	31,790	530,661	25,555	0,00	-1,73	36,231	34,496	643,610	27,088	0,00	-0,88	41,506	40,653	738,467	30,939	0,00	-0,23

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com max_it = 5.

Tabela 6.29: Influência da variação de max_it no valor de RMSE para a base Human (MCAR).

	15%			50%			80%		
max_it	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
1	44,844	72,107	1,80	48,322	77,895	2,59	59,660	103,868	-1,84
3	45,735	73,146	-0,15	49,071	76,205	1,08	59,333	104,789	-1,28
5	45,665	71,833	0,00	49,608	77,298	0,00	58,583	101,282	0,00
7	46,110	73,059	-0,98	50,036	83,920	-0,86	58,513	99,144	0,12
9	46,177	72,744	-1,12	49,741	77,182	-0,27	59,410	102,889	-1,41
11	46,369	73,349	-1,54	50,169	78,409	-1,13	58,240	94,998	0,59
13	46,474	73,233	-1,77	50,027	79,008	-0,84	58,099	94,319	0,83

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com max_it = 5.

Tabela 6.30: Influência da variação de max_it no Tempo de Execução para a base Human (MCAR).

	15%		50%		80%	
max_it	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**
1	1,70	81,37	6,95	65,93	11,70	57,34
3	4,63	49,15	14,67	28,08	20,46	25,39
5	9,11	0,00	20,40	0,00	27,42	0,00
7	9,68	-6,20	25,58	-25,40	33,51	-22,17
9	11,85	-29,98	30,29	-48,45	42,34	-54,40
11	16,26	-78,38	40,32	-97,62	45,70	-66,64
13	18,69	-105,08	48,19	-136,19	58,37	-112,83

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com max_it = 5.

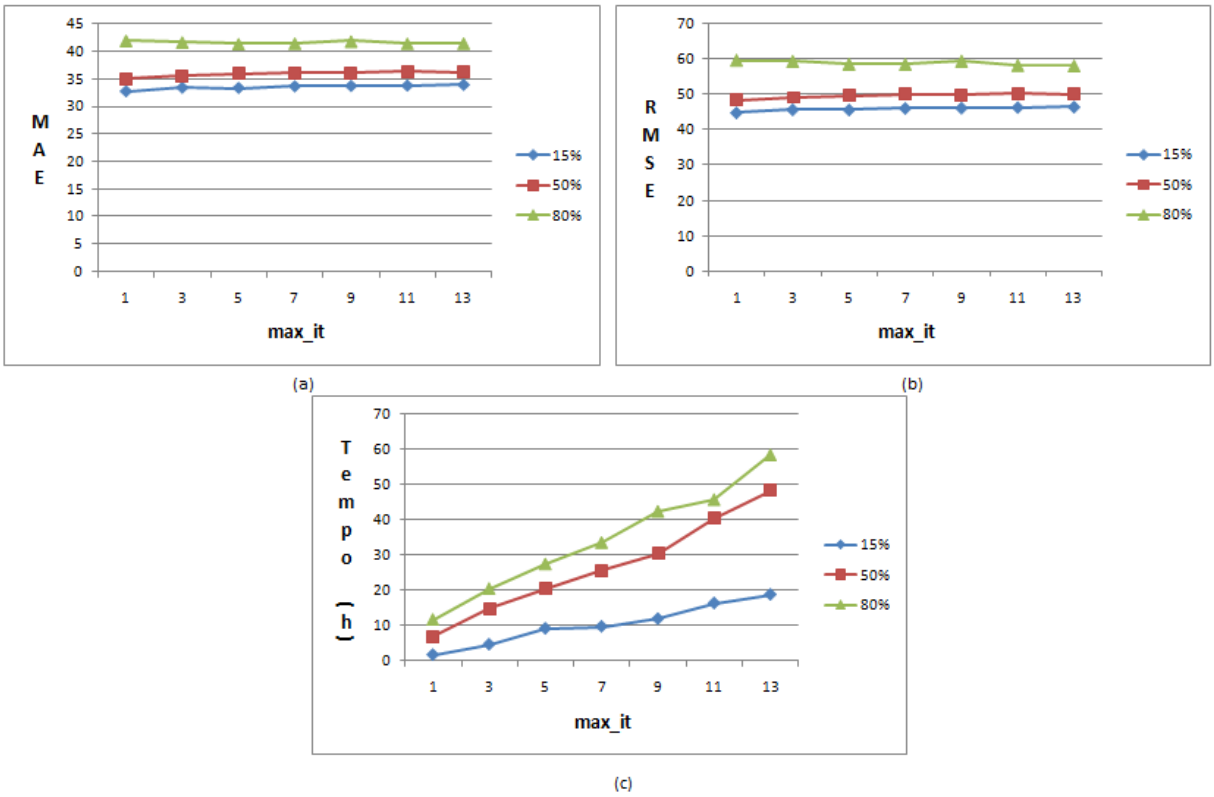


Figura 6.14: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Human (MCAR) com a variação de max_it.

6.2.3 Terceiro Experimento: a influência de n_ants

O terceiro experimento investiga a influência da variação do parâmetro n_ants no algoritmo SwarmBcluster. Os parâmetros utilizados são os mesmos apresentados na Tabela 6.24, com exceção de n_ants, o qual foi variado.

A influência da variação de n_ants no valor de MAE é mostrada na Tabela 6.31. A variação desse parâmetro influenciou muito pouco o valor de MAE. Considerando todos os testes, a maior variação não ultrapassou 0,6%. O erro padrão e a mediana também foram pouco influenciados pela variação desse parâmetro. O erro absoluto máximo não apresentou correlação com o valor de n_ants. Para todos os valores desse parâmetro, foram obtidas imputações perfeitas. Logo, em pelo menos um caso o erro absoluto foi igual a zero.

Tabela 6.31: Influência da variação de n_ants no valor de MAE para a base Human (MCAR).

	15%						50%						80%					
n_ants	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
1	33,359	31,267	523,692	25,459	0,00	-0,11	35,715	34,019	530,978	26,802	0,00	0,56	41,504	41,449	680,240	30,759	0,00	-0,23
3	33,312	31,403	512,677	25,064	0,00	0,04	35,920	34,162	499,420	26,875	0,00	-0,01	41,542	41,076	666,385	30,875	0,00	-0,32
5	33,324	31,222	532,361	25,236	0,00	0,00	35,915	34,221	502,147	26,761	0,00	0,00	41,410	41,440	857,000	30,839	0,00	0,00
7	33,391	31,241	519,175	25,553	0,00	-0,20	35,825	33,990	526,622	26,870	0,00	0,25	41,547	41,078	829,407	31,113	0,00	-0,33
9	33,393	31,464	504,180	25,225	0,00	-0,21	35,739	33,855	469,314	26,736	0,00	0,49	41,468	40,998	646,025	30,759	0,00	-0,14
11	33,412	31,409	566,748	24,917	0,00	-0,26	35,810	34,314	661,544	26,838	0,00	0,29	41,600	41,257	664,000	30,955	0,00	-0,46
13	33,520	31,635	604,249	25,247	0,00	-0,59	35,812	34,250	615,345	26,748	0,00	0,29	41,582	41,331	718,023	30,947	0,00	-0,42

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com n_ants = 5.

Tabela 6.32: Influência da variação de n_ants no valor de RMSE para a base Human (MCAR).

	15%			50%			80%		
n_ants	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
1	45,721	72,398	-0,12	49,324	77,741	0,57	58,657	98,933	-0,13
3	45,780	72,557	-0,25	49,571	77,518	0,08	58,421	96,795	0,28
5	45,665	71,833	0,00	49,608	77,298	0,00	58,583	101,282	0,00
7	45,727	72,281	-0,14	49,384	76,848	0,45	58,425	97,162	0,27
9	45,882	71,947	-0,47	49,228	75,995	0,77	58,313	95,667	0,46
11	45,857	72,815	-0,42	49,596	80,522	0,02	58,589	97,339	-0,01
13	46,091	74,747	-0,93	49,554	79,131	0,11	58,629	99,503	-0,08

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com n_ants = 5.

Tabela 6.33: Influência da variação de n_ants no Tempo de Execução para a base Human (MCAR).

	15%		50%		80%	
n_ants	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**	**Tempo (h)**	**%Melhora**
1	1,58	82,63	9,47	53,59	15,93	41,90
3	4,41	51,67	15,54	23,85	21,69	20,91
5	9,11	0,00	20,40	0,00	27,42	0,00
7	9,85	-8,13	30,18	-47,92	36,24	-32,13
9	12,67	-39,03	32,35	-58,54	38,61	-40,78
11	15,41	-69,04	38,43	-88,38	45,51	-65,95
13	18,70	-105,19	43,35	-112,50	52,05	-89,78

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com n_ants = 5.

A influência da variação de n_ants no valor de RMSE é mostrada na Tabela 6.32. A variação no valor desse parâmetro influenciou muito pouco o valor de RMSE. Considerando todos os testes, a maior variação não ultrapassou 0,95%. O desvio-padrão também foi pouco influenciado e não apresenta correlação com o valor desse parâmetro.

A influência da variação de n_ants no tempo de execução é mostrada na Tabela 6.33. Como era esperado, o aumento no valor desse parâmetro aumentou o tempo de execução.

A Figura 6.15 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução, com a variação no valor de n_ants. Vale notar que, com relação ao MAE e ao RMSE, os resultados foram bastante estáveis. Porém, o tempo de execução foi bastante influenciado pelo valor de n_ants. Nesse cenário, fica claro que n_ants = 1 foi a melhor opção.

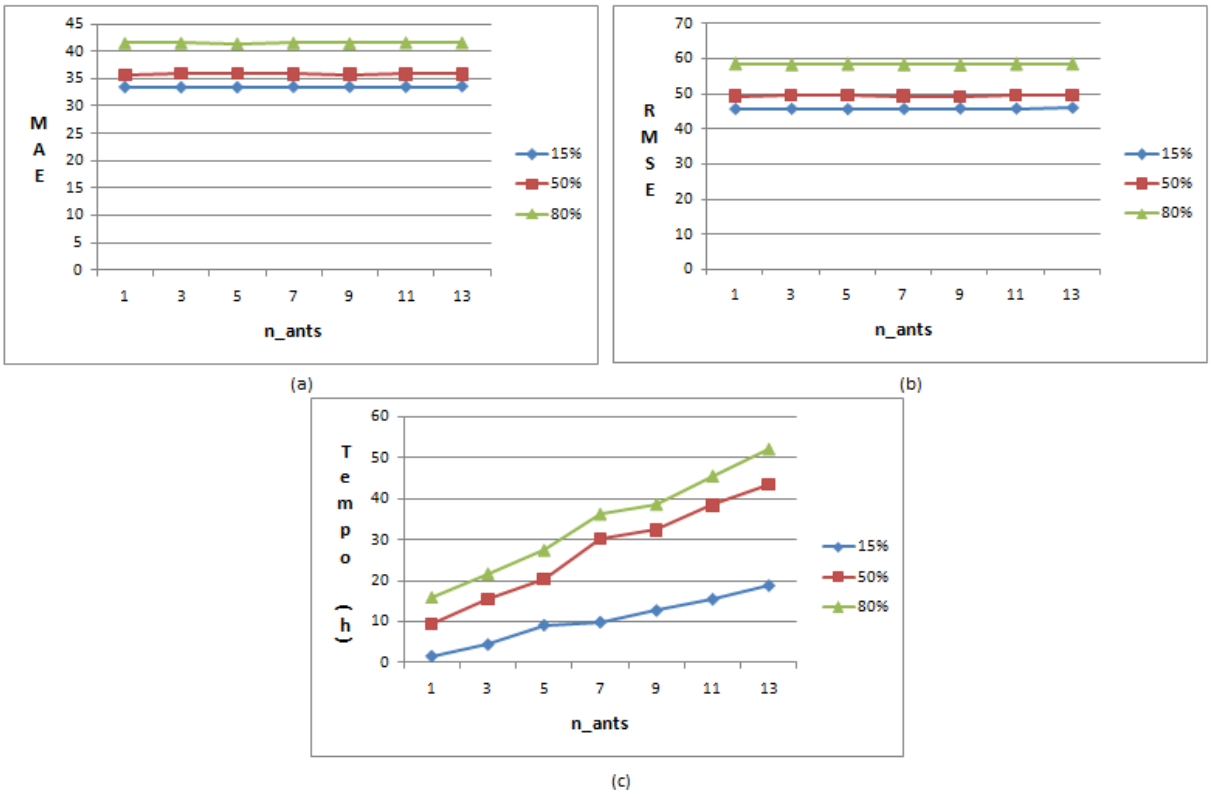


Figura 6.15: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Human (MCAR) com a variação de n_ants.

6.2.4 Quarto Experimento: o desempenho da heurística

O quarto experimento investiga o comportamento exclusivamente da heurística construtiva descrita na Seção 5.3. Para isso, foi utilizado max_it = 1 e n_ants = 1. Dessa forma, o algoritmo ACO produzirá biclusters em uma única iteração e utilizando apenas uma única formiga. Os demais parâmetros do SwarmBcluster são idênticos aos apresentados na Tabela 6.24.

Através da análise de sensibilidade paramétrica, observou-se que o parâmetro que exerceu maior influência sobre os valores de MAE e RMSE foi δ. Por isso, a heurística construtiva foi executada mais uma vez utilizando os melhores valores de δ com relação ao RMSE. Logo, para a base de dados com 15% e 50% de dados faltantes, foi utilizado δ = 600 e, para a base com 80%, foi utilizado δ = 1200. Os resultados são exibidos na Tabela 6.34.

Os testes realizados na Subseção 6.2.1, a qual analisou a influência de δ e utilizou max_it = 5 e n_ants = 5, obtiveram custos computacionais bem maiores que este experimento. Em termos de qualidade de imputação, os resultados aqui obtidos também foram bastante competitivos. Para a base de dados com 15% de dados faltantes, os valores de MAE e RMSE

foram apenas 0,51% e 1,41% piores, respectivamente. Para a base de dados com 50% de dados faltantes, os valores de MAE e RMSE foram apenas 0,72% e 1,6% piores, respectivamente. Para a base de dados com 80% de dados faltantes, o valor de MAE foi apenas 0,26% pior e o valor de RMSE foi 0,02% melhor. A grande vantagem desse experimento para aquele é que este produz imputações com uma qualidade equivalente, mas com custo computacional bem menor.

Tabela 6.34: Comportamento da heurística construtiva com o melhor δ para a base Human (MCAR).

	15%		**50%**		**80%**	
MAE	31,043	30,887	33,741	33,500	41,518	41,410
D. Padrão	30,091	29,386	32,786	31,969	41,313	41,440
Máximo	636,500	564,093	757,000	459,000	786,621	857,000
Mediana	23,165	23,220	25,096	24,967	30,920	30,839
Mínimo	0,00	0,00	0,00	0,00	0,00	0,00
RMSE	43,233	42,632	47,046	46,306	58,571	58,583
D. Padrão	77,810	71,497	81,535	72,610	98,505	101,282
Tempo (h)	1,16	21,01	2,40	43,05	8,90	27,42

Nota: Os resultados da Subseção 6.2.1 estão reproduzidos em cinza.

6.2.5 Quinto Experimento: comparação de desempenho com respeito ao mecanismo MCAR

Este experimento visa avaliar o desempenho do SwarmBcluster com respeito ao mecanismo MCAR. Para isso, seu desempenho foi comparado ao KNNImpute e ao rSVD, descritos no Capítulo 5.

O desempenho dos três algoritmos foi avaliado em testes realizados com doze conjuntos de dados faltantes produzidos artificialmente, segundo o mecanismo MCAR, com taxas de 2%, 5%, 10%, 15%, 20%, 30%, 40%, 50%, 60%, 70%, 80% e 90%. Na primeira etapa desse experimento, os parâmetros utilizados para o SwarmBcluster foram os apresentados na Tabela 6.24. Em seguida, para as bases de 15% e 50%, as quais foram utilizadas nos experimentos de sensibilidade paramétrica, será apresentado o ganho de desempenho do SwarmBcluster com relação ao KNNImpute e ao rSVD. A base de 80% de dados faltantes também foi utilizada nos experimentos de sensibilidade paramétrica, mas não foi possível melhorar a parametrização apresentada na Tabela 6.24.

Tabela 6.35: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Human (MCAR).

	SwarmBcluster					KNNImpute					rSVD					% Melhora	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
2	34,241	31,625	220,927	25,730	0,04	44,910	45,593	407,555	32,312	0,04	50,018	52,058	420,509	35,727	0,01	31,16	46,08
5	33,959	32,263	383,025	25,796	0,00	45,500	44,701	582,455	33,895	0,00	52,065	53,217	537,814	37,750	0,00	33,98	53,32
10	33,692	32,304	405,415	25,138	0,01	45,032	44,855	548,769	32,496	0,01	49,978	52,574	680,810	35,520	0,00	33,66	48,34
15	33,324	31,222	532,361	25,236	0,00	45,197	43,799	540,497	33,476	0,00	50,219	51,648	572,884	35,918	0,00	35,63	50,70
20	33,256	31,284	411,116	25,029	0,00	44,750	44,566	506,902	32,884	0,00	51,959	53,957	559,738	36,424	0,00	34,56	56,24
30	33,904	32,118	477,550	25,230	0,00	45,680	45,533	554,516	33,348	0,00	51,546	54,241	668,485	36,280	0,00	34,74	52,04
40	34,835	33,286	517,208	26,034	0,00	46,469	46,625	595,627	33,840	0,00	52,235	55,616	636,260	36,432	0,00	33,40	49,95
50	35,915	34,221	502,147	26,761	0,00	47,140	46,492	598,255	34,546	0,00	54,733	59,790	855,000	37,867	0,00	31,25	52,39
60	37,250	35,728	920,000	28,077	0,00	48,591	47,918	638,946	35,691	0,00	58,505	64,853	802,553	39,858	0,00	30,44	57,06
70	39,043	38,156	822,000	29,226	0,00	50,367	49,062	609,353	37,231	0,00	59,762	67,713	884,605	40,056	0,00	29,01	53,07
80	41,410	41,440	857,000	30,839	0,00	53,159	51,237	686,587	39,438	0,00	76,929	103,938	1245,000	44,612	0,00	28,37	85,78
90	45,449	46,235	822,959	34,024	0,00						89,578	111,876	1141,000	52,246	0,00		97,09

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.35 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. O algoritmo KNNImpute não foi capaz de tratar a base de dados com 90% de dados faltantes. Em todos os testes, o SwarmBcluster foi superior aos outros algoritmos. O SwarmBcluster obteve um ganho de desempenho máximo de 35,63% em relação ao KNNImpute e de 97,09% em relação ao rSVD. Analisando o valor de MAE juntamente com o valor do desvio-padrão, o SwarmBcluster também obteve desempenho superior aos outros dois algoritmos. Para o valor máximo do erro absoluto, o SwarmBcluster foi superior ao KNNImpute em quase todos os casos, perdendo apenas em 3 casos, nos quais a porcentagem de dados faltantes é superior a 60%. Com relação ao rSVD, o SwarmBcluster perdeu apenas para o caso com 60% de dados faltantes. Os resultados para a mediana do erro absoluto se comportaram da mesma maneira que os resultados para o valor de MAE. Em todos os testes realizados neste experimento, os três algoritmos conseguiram estimar valores idênticos, ou muito próximos, aos valores reais da base de dados, como é possível observar através do valor mínimo do erro absoluto.

Tabela 6.36: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Human (MCAR).

	SwarmBcluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
2	46,611	65,762	63,997	103,263	72,193	116,810	37,30	54,88
5	46,841	72,947	63,784	103,403	74,450	120,119	36,17	58,94
10	46,677	72,436	63,560	101,552	72,538	123,012	36,17	55,40
15	45,665	71,833	62,937	99,016	72,038	118,881	37,82	57,75
20	45,658	70,864	63,156	102,346	74,907	122,679	38,33	64,06
30	46,701	72,328	64,498	103,418	74,827	125,761	38,11	60,22
40	48,181	76,501	65,827	106,712	76,300	127,546	36,62	58,36
50	49,608	77,298	66,210	104,931	81,058	139,417	33,47	63,40
60	51,615	85,654	68,244	109,128	87,343	152,084	32,22	69,22
70	54,591	91,670	70,313	109,979	90,313	159,500	28,80	65,44
80	58,583	101,282	73,832	114,630	129,310	238,958	26,03	120,73
90	64,833	113,246			143,320	245,030		121,06

Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.36 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Em todos os testes realizados, o SwarmBcluster foi superior aos outros dois algoritmos. O SwarmBcluster foi até 38,33% melhor que o KNNImpute e foi até

121,06% melhor que o rSVD. Analisando o valor de RMSE juntamente com o valor do desvio-padrão, o SwarmBcluster também foi melhor que os outros dois algoritmos.

A Tabela 6.37 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base de dados Human (MCAR). Como já foi comentado, o SwarmBcluster tem um custo computacional maior que os outros algoritmos.

Tabela 6.37: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao Tempo de Execução para a base Human (MCAR).

	SwarmBcluster		**KNNImpute**	**rSVD**
%	**Tempo (s)**	**Tempo (h)**	**Tempo (s)**	**Tempo (s)**
2	6.009	1,67	28,06	68,82
5	13.489	3,75	34,63	66,63
10	22.113	6,14	36,88	70,21
15	32.811	9,11	45,53	62,38
20	31.805	8,83	54,00	41,43
30	50.008	13,89	69,27	89,85
40	59.673	16,58	83,93	69,30
50	73.447	20,40	96,97	55,89
60	87.572	24,33	108,44	74,97
70	89.423	24,84	117,82	75,65
80	98.729	27,42	125,48	127,05
90	72.211	20,06		143,91

A Figura 6.16 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Através dela, é possível verificar que os comportamentos de MAE e RMSE são bastante parecidos. O rSVD foi o algoritmo que apresentou resultados mais instáveis. O comportamento do SwarmBcluster e do KNNImpute são semelhantes, mas os resultados do SwarmBcluster foram visivelmente melhores que os do KNNImpute. Esses dois algoritmos foram bem estáveis até 50% de dados faltantes. Com mais de 50% de dados faltantes, a piora na qualidade das imputações tem um leve crescimento.

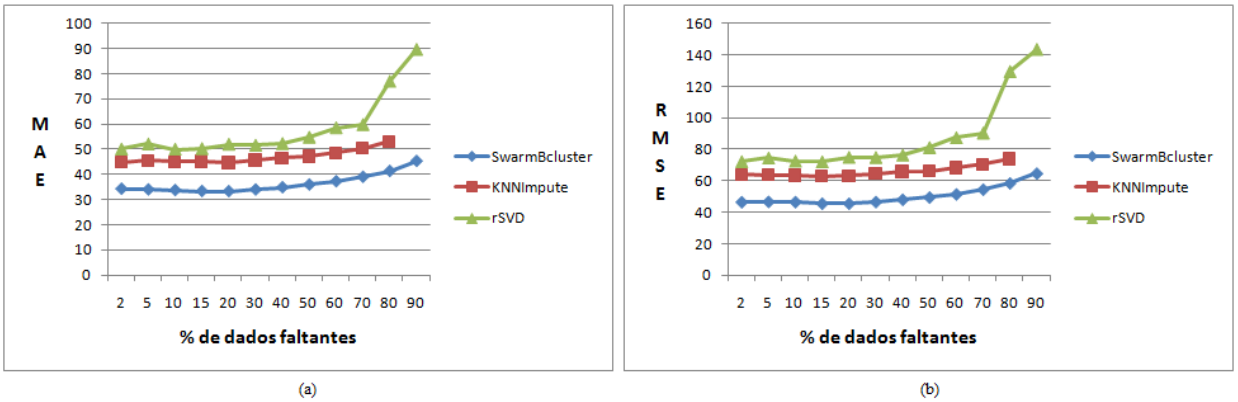


Figura 6.16: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD quando aplicados à base de dados Human (MCAR).

A Figura 6.17 mostra o comportamento do valor de MAE, RMSE e do tempo de execução para o algoritmo SwarmBcluster. O tempo de execução apresentou um comportamento diferente do exibido pelo MAE e RMSE. Ele foi crescente até 80% de dados faltantes. Com 90% de dados faltantes, o tempo de execução caiu levemente em relação aos casos com 60% a 80%.

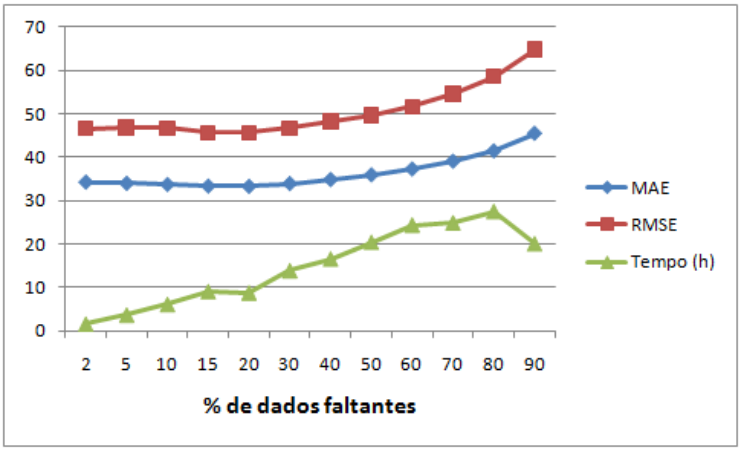


Figura 6.17: Comportamento do valor de MAE, RMSE e do Tempo de Execução para o algoritmo SwarmBcluster quando aplicado à base de dados Yeast (MCAR).

A Figura 6.18 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD para as bases de dados com 15%, 50% e 80% de dados faltantes. Para todas essas bases de dados, o SwarmBcluster apresentou erros absolutos mais concentrados e com menor variação que os demais algoritmos.

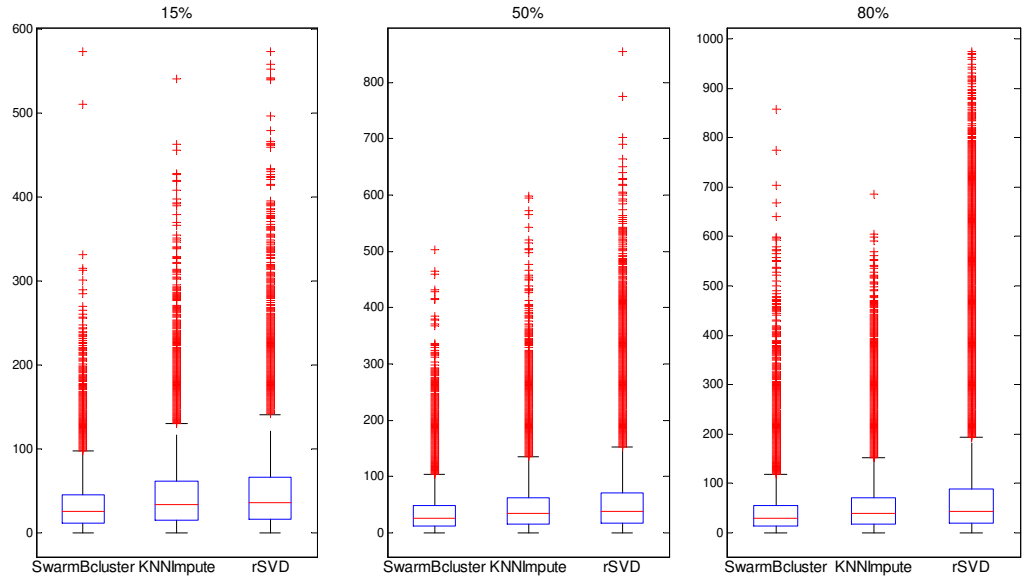


Figura 6.18: Erros absolutos produzidos por SwarmBcluster, KNNImpute e rSVD para a base de dados Yeast (MCAR) com 15%, 50% e 80% de dados faltantes.

Através da análise de sensibilidade paramétrica, observou-se que a variação de δ foi o experimento que trouxe maiores ganhos em termos de MAE e RMSE, ou seja, em termos de qualidade de imputação. Por isso, a comparação de desempenho entre o SwarmBcluster, o KNNImpute e o rSVD é exibida novamente para as bases de dados com 15% e 50% de dados faltantes, as quais participaram da análise de sensibilidade paramétrica e obtiveram um melhor desempenho após essa análise. O melhor valor de δ encontrado foi 600. Os outros parâmetros utilizados foram iguais àqueles apresentados na Tabela 6.24.

A Tabela 6.38 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. Para a base de dados com 15% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 35,63% para 46,33% e com relação ao rSVD houve um aumento de 50,7% para 62,59%. Além disso, os valores do desvio-padrão e da mediana são menores neste teste do que os obtidos com δ = 1200, apenas o valor máximo do erro absoluto é um pouco maior. Para a base de dados com 50% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 31,25% para 40,72% e com relação ao rSVD aumentou de 52,39% para 63,38%.

Tabela 6.38: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para as bases Human (MCAR) utilizadas nos experimentos de sensibilidade paramétrica.

	SwarmBcluster					KNNImpute					rSVD					%MelAnter		%MelAtual	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**	**1 vs. 2**	**1 vs. 3**
15	30,887	29,386	564,093	23,220	0,00	45,197	43,799	540,497	33,476	0,00	50,219	51,648	572,884	35,918	0,00	35,63	50,70	46,33	62,59
50	33,500	31,969	459,000	24,967	0,00	47,140	46,492	598,255	34,546	0,00	54,733	59,790	855,000	37,867	0,00	31,25	52,39	40,72	63,38

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto, %MelAnter → porcentagem de melhora anterior e %MelAtual → porcentagem de melhora atual. A porcentagem de melhora indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.39 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Para a base de dados com 15% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 37,82% para 47,63% e com relação ao rSVD aumentou de 57,75% para 68,98%. Para a base de dados com 50% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 33,47% para 42,98% e com relação ao rSVD aumentou de 63,4% para 75,05%. Para as duas bases de dados, o valor do desvio-padrão é menor neste teste do que o obtido com $\delta = 1200$.

Tabela 6.39: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para as bases Human (MCAR) utilizadas nos experimentos de sensibilidade paramétrica.

	SwarmBcluster		**KNNImpute**		**rSVD**		**%MelAnter**		**%MelAtual**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**	**1 vs. 2**	**1 vs. 3**
15	42,632	71,497	62,937	99,016	72,038	118,881	37,82	57,75	47,63	68,98
50	46,306	72,610	66,210	104,931	81,058	139,417	33,47	63,40	42,98	75,05

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, %MelAnter → porcentagem de melhora anterior e %MelAtual → % melhora de atual. A porcentagem de melhora indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

6.2.6 Sexto Experimento: comparação de desempenho com respeito ao mecanismo MAR

O sexto experimento visa comparar o desempenho do SwarmBcluster com relação ao mecanismo MAR. Para isso, seus resultados foram comparados aos resultados obtidos pelo KNNImpute e pelo rSVD, descritos no Capítulo 5.

Para realizar os testes, foram produzidos quatro casos artificiais de dados faltantes, a partir da base Human, da seguinte maneira:

1. O atributo 32 foi escolhido aleatoriamente para ser o motivo da ausência dos dados;
2. Os atributos 22, 25, 34 e 40 foram escolhidos aleatoriamente para apresentar dados faltantes associados a uma dada probabilidade, sempre que o valor do atributo 32 fosse maior que sua média.
3. As probabilidades escolhidas foram: 20%, 40%, 60% e 80%, gerando, portanto, os quatro casos artificiais de dados faltantes.

Tabela 6.40: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Human (MAR).

	SwarmBCluster					KNNImpute					rSVD					% Melhora	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
20	30,944	27,259	236,282	24,243	0,05	34,527	33,902	226,809	25,370	0,156	38,977	34,983	223,458	30,979	0,070	11,58	25,96
40	30,991	29,344	255,070	23,435	0,02	34,929	33,271	280,277	26,471	0,006	39,282	35,132	285,711	31,056	0,063	12,71	26,75
60	30,468	28,318	265,580	22,732	0,02	34,906	33,042	279,030	26,476	0,050	39,807	35,002	285,869	32,397	0,010	14,57	30,65
80	32,410	30,550	340,316	24,374	0,04	37,076	35,224	250,649	28,656	0,018	40,279	36,432	291,202	32,051	0,004	14,40	24,28

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

Os parâmetros utilizados foram os mesmos dos apresentados na Tabela 6.24.

A Tabela 6.40 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. O SwarmBcluster foi superior aos outros dois algoritmos. Ele foi até 14,57% melhor que o KNNImpute e até 30,65% melhor que o rSVD. O SwarmBcluster também produziu valores menores para a mediana e o desvio-padrão. Para o caso de 40% e 60%, o SwarmBcluster obteve melhores valores de maior erro absoluto. Em todos os testes realizados neste experimento, os três algoritmos conseguiram estimar valores idênticos, ou muito próximos, aos valores reais da base de dados, como se pode observar através do valor mínimo do erro absoluto.

A Tabela 6.41 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Novamente, o SwarmBcluster foi superior aos demais algoritmos. Ele foi até 17,34% melhor que o KNNImpute e 27,43% melhor que o rSVD. O SwarmBcluster também produziu valores menores para o desvio-padrão.

Tabela 6.41: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Human (MAR).

	SwarmBCluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
20	41,238	61,230	48,389	73,632	52,374	74,889	17,34	27,00
40	42,679	65,164	48,240	74,366	52,700	76,427	13,03	23,48
60	41,595	62,248	48,065	74,211	53,007	77,454	15,55	27,43
80	44,539	69,010	51,140	77,731	54,311	79,988	14,82	21,94

Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

Tabela 6.42: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base Human (MAR).

	SwarmBCluster		**KNNImpute**	**rSVD**
%	**Tempo (s)**	**Tempo (min)**	**Tempo (s)**	**Tempo (s)**
20	516,43	8,61	20,41	64,21
40	819,22	13,65	20,42	64,61
60	793,08	13,22	20,92	64,61
80	1143,71	19,06	22,28	64,75

A Tabela 6.42 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução. Como era esperado, o SwarmBcluster tem um custo computacional maior que os outros algoritmos.

A Figura 6.19 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Apesar de o algoritmo rSVD ter sido o mais estável, ele obteve os piores resultados. O SwarmBcluster e o KNNImpute foram estáveis até 60% de dados faltantes. Com 80% de dados faltantes, esses dois algoritmos apresentaram uma pequena piora na qualidade da imputação. O SwarmBcluster foi visivelmente melhor que os outros dois algoritmos.

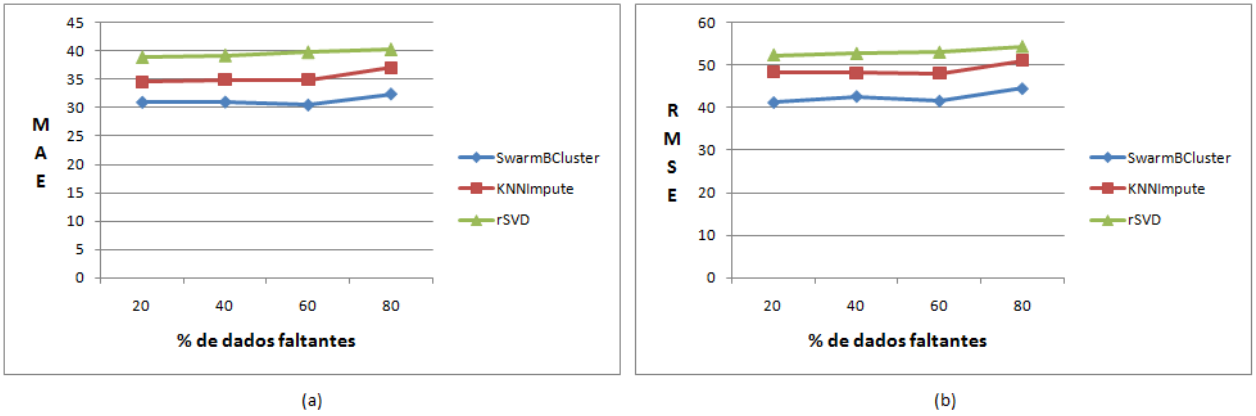


Figura 6.19: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD para a base de dados Human (MAR).

A Figura 6.20 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD, para as bases de dados com 40% e 80% de dados faltantes. Para ambos os casos, o SwarmBcluster foi visivelmente melhor que o rSVD e levemente superior ao KNNImpute. As diferenças entre o SwarmBcluster e o KNNImpute foram mais acentuadas para o caso com 80% de dados faltantes.

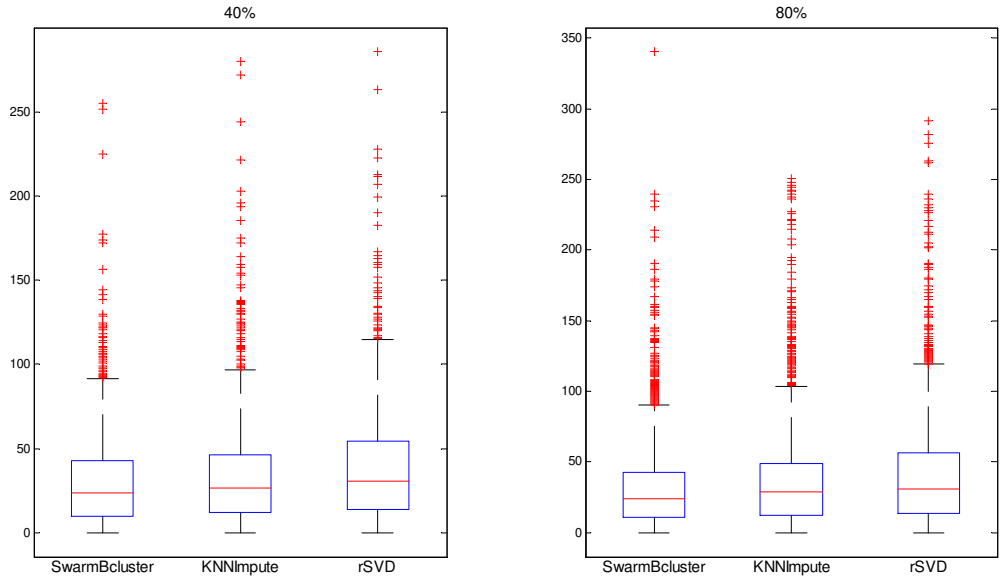


Figura 6.20: Erros absolutos produzidos por SwarmBcluster, KNNImpute e rSVD para a base de dados Human (MAR) com 40% e 80% de dados faltantes.

6.2.7 Sétimo Experimento: comparação de desempenho com respeito ao mecanismo MNAR

O sétimo experimento visa comparar o desempenho do SwarmBcluster com relação ao mecanismo MNAR. Para isso, seus resultados foram comparados aos resultados obtidos pelo KNNImpute e rSVD, descritos no Capítulo 5.

Para realizar os testes, foram produzidos quatro casos artificiais, da seguinte maneira:

1. O atributo 34 foi escolhido aleatoriamente para ser o motivo da ausência dos dados. Esse atributo tem certa probabilidade de possuir dados faltantes quando o seu valor é maior que sua média.
2. As probabilidades escolhidas foram 20%, 40%, 60% e 80%, gerando, portanto, as quatro bases de dados artificiais.

Os parâmetros utilizados são os mesmos apresentados na Tabela 6.24, com exceção de δ. O valor utilizado foi $\delta = 600$.

Tabela 6.43: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Human (MNAR).

	SwarmBCluster					KNNImpute					rSVD					% Melhora	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
20	37,666	33,438	234,000	33,821	0,47	42,867	40,172	256,797	33,093	0,92	61,754	56,561	290,208	45,313	0,21	13,81	63,95
40	44,978	41,175	364,539	35,259	1,90	44,144	41,543	327,392	35,850	0,06	69,470	60,215	449,738	55,429	0,36	-1,85	54,46
60	45,409	41,228	334,556	36,144	0,29	41,895	41,576	329,052	31,223	0,11	75,965	75,260	502,505	64,809	0,93	-7,74	67,29
80	57,972	51,070	518,905	45,999	0,00	51,070	44,780	292,970	42,557	0,34	98,016	91,267	589,206	76,457	0,06	-11,91	69,07

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.43 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. Para todos os casos, o SwarmBcluster produziu melhores valores de MAE em relação ao rSVD, mas para apenas um caso ele foi melhor que o KNNImpute. Nesse caso, ele foi 13,81% melhor que o KNNImpute. O SwarmBcluster também produziu resultados melhores que o rSVD em relação ao desvio-padrão e a mediana. Com relação ao KNNImpute, o SwarmBcluster produziu resultados melhores para o desvio-padrão para os três primeiros casos e produziu resultado melhor para a mediana apenas para o caso com 40% de dados faltantes. O SwarmBcluster também obteve melhor valor do erro absoluto máximo do que o rSVD obteve, mas, apenas para o caso com 20% de dados faltantes, o SwarmBcluster foi melhor que o KNNImpute nesse quesito. Todos os algoritmos apresentaram bons valores para o erro absoluto mínimo.

Tabela 6.44: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Human (MNAR).

	SwarmBCluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
20	50,367	79,424	58,748	87,558	83,742	116,163	16,64	66,26
40	60,979	102,586	60,618	93,388	91,935	129,135	-0,59	50,77
60	61,332	97,719	59,023	93,876	106,934	150,567	-3,76	74,35
80	77,258	127,333	67,922	98,136	133,928	182,386	-12,08	73,35

Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

Tabela 6.45: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base Human (MNAR).

	SwarmBCluster		**KNNImpute**	**rSVD**
%	**Tempo (s)**	**Tempo (min)**	**Tempo (s)**	**Tempo (s)**
20	370,53	6,18	18,99	66,11
40	839,91	14,00	19,19	69,85
60	874,18	14,57	19,42	37,66
80	1178,87	19,65	19,59	38,47

A Tabela 6.44 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Para todos os casos, o SwarmBcluster obteve valores melhores de RMSE em relação ao rSVD mas, quando comparado ao KNNImpute, ele obteve melhor resultado em apenas um caso.

A Tabela 6.45 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução. Como era esperado, o SwarmBcluster tem um custo computacional maior que os outros algoritmos. Mas o tempo de execução do SwarmBcluster não passou de 20 minutos. Como os dados faltantes são apenas em relação a um atributo, mesmo com uma alta porcentagem de dados faltantes, a quantidade absoluta é muito menor que nos testes feitos com o mecanismo MCAR.

A Figura 6.21 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Através dela, é possível verificar que os comportamentos de MAE e RMSE foram bastante parecidos. O SwarmBcluster e o KNNImpute apresentaram comportamento e desempenho parecidos. Além do mais, eles também foram bastante superiores ao rSVD.

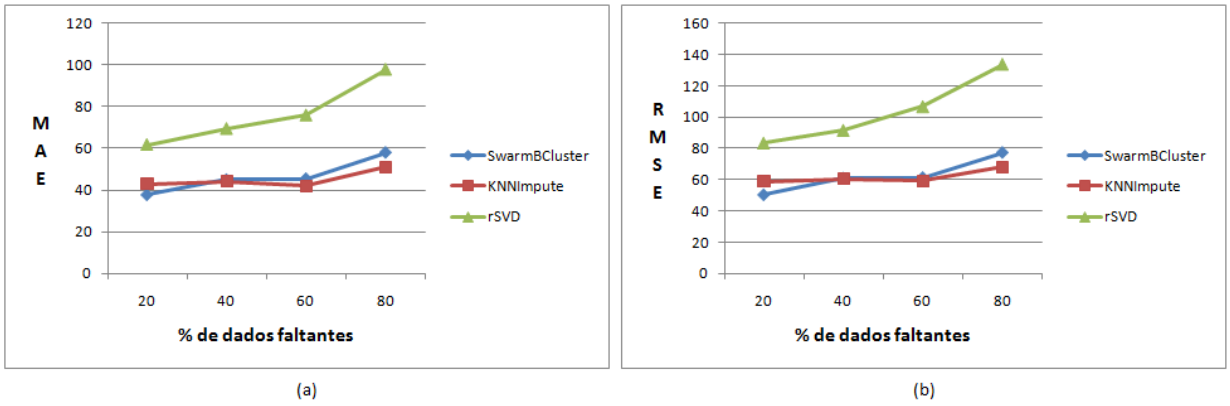


Figura 6.21: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD para a base de dados Human (MNAR).

A Figura 6.22 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD para as bases de dados com 40% e 80% de dados faltantes. Os erros absolutos obtidos pelo SwarmBcluster e pelo KNNImpute foram bastante parecidos e também foram melhores que os erros absolutos obtidos pelo rSVD.

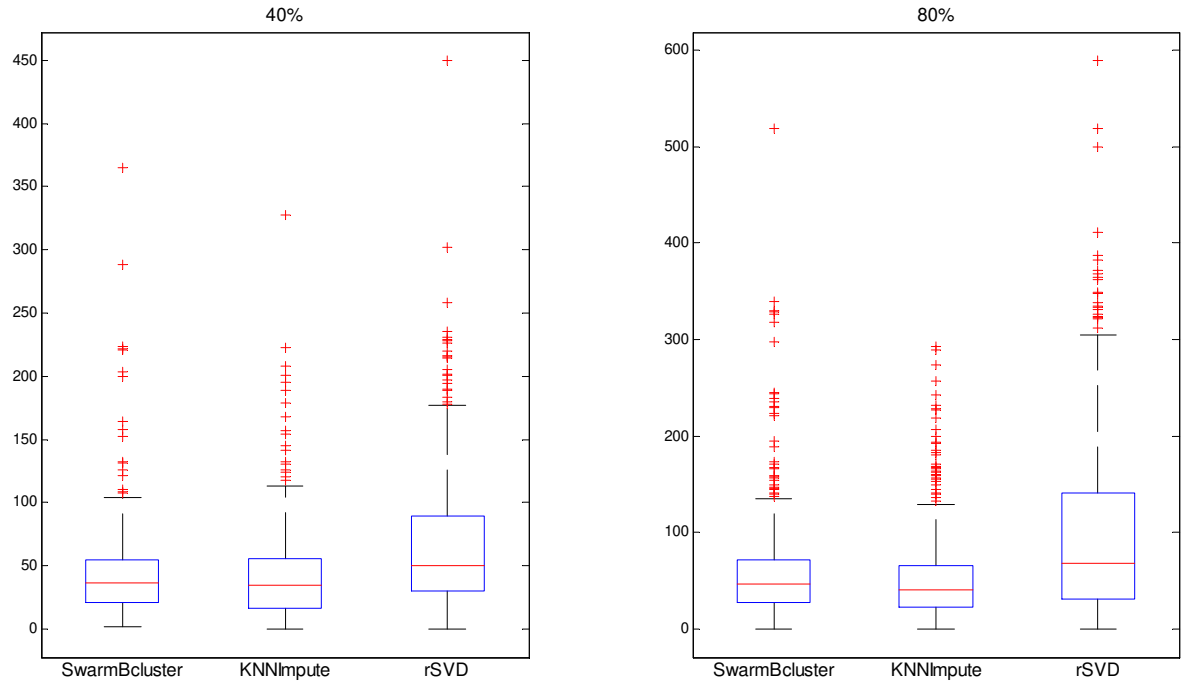


Figura 6.22: Erros absolutos produzidos pelo SwarmBcluster, KNNImpute e rSVD para a base de dados Human (MNAR) com 40% e 80% de dados faltantes.

6.3 Base de dados Jester

Nesta Seção, serão descritos os experimentos realizados com a base de dados Jester. Esses experimentos foram realizados em um computador Intel Core^{tm}2 Quad Q6600 @ 2,40 Ghz, 2 GB de RAM, utilizando apenas um único *core* para cada experimento.

Tabela 6.46: Parâmetros-base utilizados pelo SwarmBcluster nos experimentos com a base de dados Jester.

Parâmetro	**Valor**
δ	6
max_it	5
n_ants	5
α	1
β	3
ρ	0,2
col_min	7
row_min	100
max_vars	2500

Os parâmetros-base para os experimentos que foram realizados são listados a seguir. O parâmetro utilizado para o KNNImpute foi $K = 15$. Como foi mencionado na Seção 5.1, para encontrar um valor de K apropriado para os experimentos deste trabalho, foram realizados alguns experimentos preliminares. Os parâmetros utilizados para o rSVD são os mesmos que foram utilizados nos experimentos com a base de dados Yeast, apresentados na Tabela 6.1. Os parâmetros utilizados para o SwarmBcluster são apresentados na Tabela 6.46. Os valores dos parâmetros foram baseados em DE FRANÇA (2010), com exceção de *col_min*. Como nos outros experimentos, o valor utilizado para esse parâmetro foi 7.

Assim como foi realizado com as bases de dados Yeast e Human, a análise da influência de δ, exibida a seguir, foi realizada com bases de dados que possuem 15%, 50% e 80% de dados faltantes, segundo o mecanismo MCAR.

6.3.1 Primeiro Experimento: a influência de δ

O primeiro experimento investiga a influência da variação do parâmetro δ no algoritmo SwarmBcluster. Os parâmetros utilizados para o SwarmBcluster são os mesmos apresentados na Tabela 6.46, com exceção de δ, o qual foi variado.

A influência da variação de δ no valor de MAE é mostrada na Tabela 6.47.

Para a base com 15% de dados faltantes, a diminuição no valor de δ trouxe resultados melhores para o valor de MAE e do desvio-padrão, enquanto o aumento no valor de δ trouxe resultados piores. Com $\delta = 2$ e $\delta = 8$, o resultado de MAE foi, respectivamente, 1,93% melhor e 1,71% pior que com $\delta = 6$. Os menores valores de δ também apresentaram os menores valores para o máximo erro absoluto. A mediana se comportou de forma idêntica ao valor de MAE e do desvio-padrão, isto é, quanto maior o valor de δ, pior o valor da mediana. Para todos os valores de δ, foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Tabela 6.47: Influência da variação de δ no valor de MAE para a base Jester (MCAR).

	15%						50%						80%					
δ	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**%Mel**
2	2,818	2,628	18,806	2,017	0,00	1,93	2,866	2,608	19,630	2,121	0,00	2,09	3,119	2,818	23,173	2,338	0,00	-1,17
4	2,839	2,672	18,873	2,028	0,00	1,20	2,881	2,632	18,922	2,115	0,00	1,57	3,030	2,738	19,420	2,246	0,00	1,72
6	2,873	2,682	19,419	2,070	0,00	0,00	2,927	2,670	19,022	2,149	0,00	0,00	3,083	2,788	19,501	2,284	0,00	0,00
8	2,922	2,706	19,120	2,131	0,00	-1,71	2,981	2,699	19,270	2,212	0,00	-1,85	3,182	2,838	19,263	2,383	0,00	-3,21

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de MAE obtido com δ = 6.

Os testes com a base com 50% de dados faltantes apresentaram um comportamento semelhante aos testes com a base de 15%, ou seja, valores menores de δ geraram melhores resultados. Com δ = 2 e δ = 8, o resultado de MAE foi, respectivamente, 2,09% melhor e 1,85% pior que com δ = 6. O valor máximo do erro absoluto não apresentou correlação com o valor de δ. O desvio-padrão e a mediana se comportaram de forma idêntica ao valor de MAE. Para todos os valores de δ, foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Novamente, o melhor valor de δ para a base de dados com 80% de dados faltantes foi o dobro do melhor valor de δ para as bases de dados com 15% e 50% de dados faltantes. O valor de δ = 4 produziu um valor de MAE 1,72% melhor que com δ = 6. Mesmo analisando o valor de δ juntamente com o desvio-padrão, o melhor valor encontrado foi δ = 4. O valor máximo do erro absoluto apresentou correlação negativa com o valor de δ. A mediana apresentou o mesmo comportamento que o apresentado pelo valor de MAE. Como nos testes com as bases com 15% e 50% de dados faltantes, para todos os valores de δ foram obtidas imputações perfeitas. Logo, em pelo menos um caso, o erro absoluto foi igual a zero.

Tabela 6.48: Influência da variação de δ no valor de RMSE para a base Jester (MCAR).

	15%			**50%**			**80%**		
δ	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**	**RMSE**	**D.Pad**	**%Mel**
2	3,853	5,248	1,97	3,875	5,244	2,20	4,204	5,744	-1,13
4	3,898	5,388	0,81	3,902	5,300	1,51	4,084	5,527	1,75
6	3,930	5,433	0,00	3,962	5,383	0,00	4,157	5,618	0,00
8	3,983	5,497	-1,34	4,021	5,447	-1,50	4,264	5,697	-2,58

Nota: Os significados das abreviações são: D.Pad → desvio-padrão e %Mel → porcentagem de melhora. A porcentagem de melhora é calculada com base no valor de RMSE obtido com δ = 6.

A influência da variação de δ no valor de RMSE é mostrada na Tabela 6.48.

Para a base com 15% de dados faltantes, a diminuição no valor de δ trouxe resultados melhores para o valor de RMSE, enquanto o aumento no valor de δ trouxe resultados piores. Com δ = 2 e δ = 8, o RMSE foi, respectivamente, 1,97% melhor e 1,34% pior do que com δ = 6. O valor do desvio-padrão também apresenta correlação positiva com o valor de δ.

Os testes com a base com 50% de dados faltantes tiveram um comportamento semelhante aos dos testes com a base com 15%. Com $\delta = 2$ e $\delta = 8$, o resultado de RMSE foi, respectivamente, 2,2% melhor e 1,5% pior que com $\delta = 6$. O valor do desvio-padrão também apresenta correlação positiva com o valor de δ.

Nos testes realizados com a base de dados com 80% de dados faltantes, o melhor valor de δ foi 4, justamente o dobro do melhor valor de δ para as bases de dados com 15% e 50% de dados faltantes. O resultado obtido com $\delta = 4$ foi 1,75% melhor que com $\delta = 6$ e também obteve o melhor valor para o desvio-padrão.

Tabela 6.49: Influência da variação de δ no tempo em dias de execução para a base Jester (MCAR).

	15%		50%		80%	
δ	**Tempo (d)**	**%Melhora**	**Tempo (d)**	**%Melhora**	**Tempo (d)**	**%Melhora**
2	3,58	-301,24	7,32	-41,55	11,97	-27,09
4	0,72	19,71	5,20	-0,61	8,07	14,36
6	0,89	0,00	5,17	0,00	9,42	0,00
8	1,16	-30,25	5,73	-10,77	13,77	-46,15

Nota: A porcentagem de melhora é calculada com base no tempo de execução obtido com $\delta = 6$.

A influência da variação de δ no tempo de execução é mostrada na Tabela 6.49. Para as bases de dados com 15% e 50% de dados faltantes, o custo computacional da execução do programa com o melhor valor de δ (em termos de qualidade de imputação) foi consideravelmente maior que o custo obtido pelos outros valores de δ testados. Já para a base de dados com 80% de dados faltantes, o melhor valor de δ em termos de qualidade de imputação também é o melhor valor de δ em termos de custo computacional.

A Figura 6.23 exibe o comportamento de (a) MAE, (b) RMSE e (c) tempo de execução, com a variação no valor de δ. Vale notar que, com relação ao MAE e ao RMSE, os experimentos realizados para os casos com 15% e 50% de dados faltantes apresentaram um comportamento semelhante. Ambos, MAE e RMSE, pioraram com o aumento no valor de δ. Já para a base de dados com 80% de dados faltantes, os valores de MAE e RMSE crescem com $\delta < 4$ ou $\delta > 4$. Em relação ao custo computacional, para as bases de dados com 15% e 50% de dados faltantes, o custo computacional tende a se estabilizar com $\delta > 4$. Já para a base de dados com 80% de dados

faltantes, o comportamento do custo computacional foi parecido com o comportamento de MAE e RMSE.

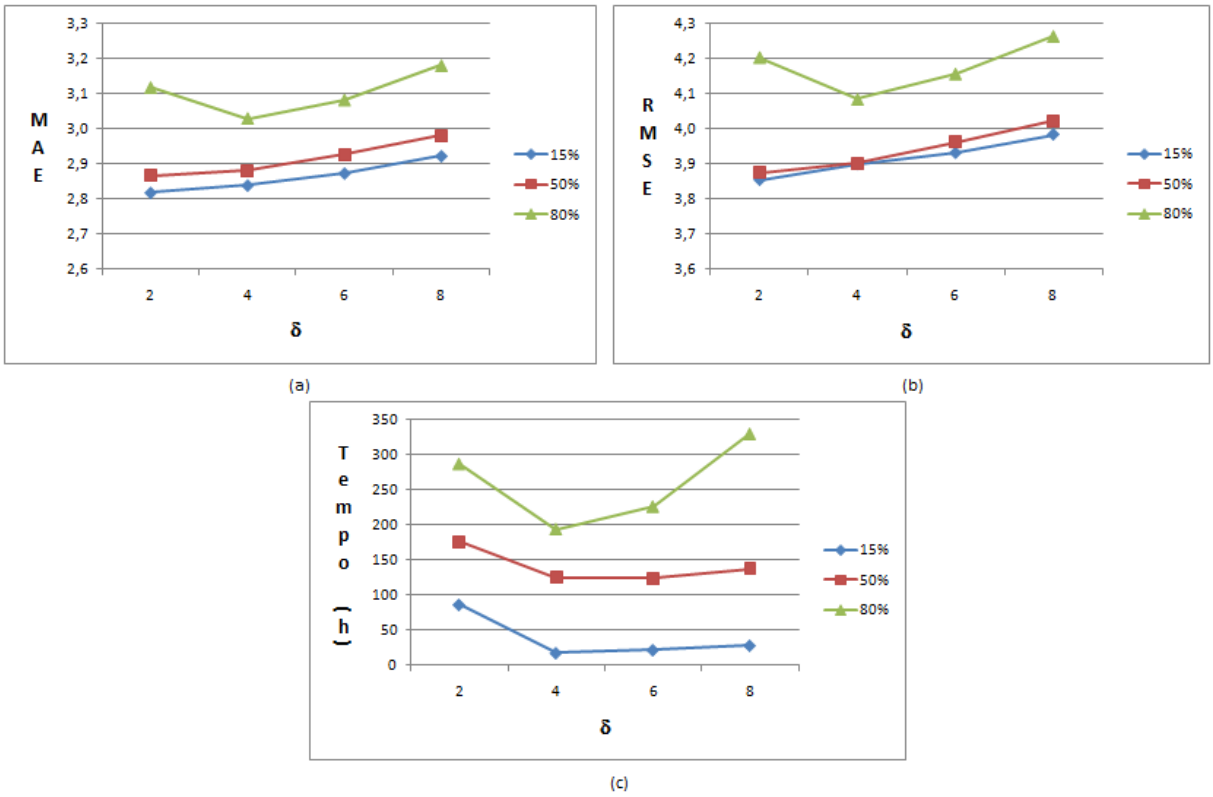


Figura 6.23: Comportamento de (a) MAE, (b) RMSE e (c) Tempo de Execução para a base de dados Jester (MCAR) com a variação de δ.

A Figura 6.24 mostra o comportamento de RMSE e do tempo de execução com a variação de δ. Para todas as bases de dados, o valor de RMSE se manteve estável com a variação de δ. O mesmo não aconteceu com o custo computacional. Logo, pensando no custo computacional, uma boa opção para o valor de δ seria 4 para todas as bases de dados.

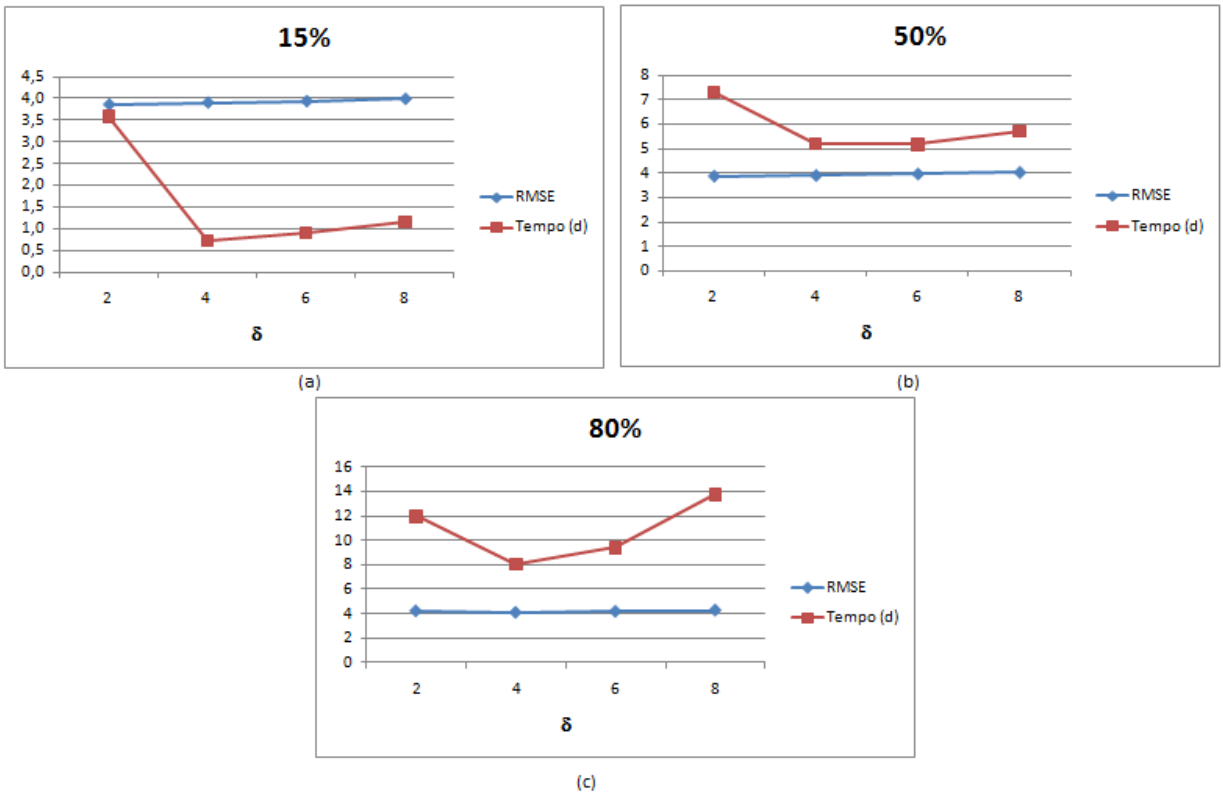


Figura 6.24: Comportamento de RMSE e do tempo de execução com a variação de δ para as bases de dados Jester (MCAR) com (a) 15%, (b) 50% e (c) 80% de dados faltantes.

6.3.2 Segundo Experimento: comparação de desempenho com respeito ao mecanismo MCAR

Este experimento visa avaliar o desempenho do SwarmBcluster com respeito ao mecanismo MCAR. Para isso, seu desempenho foi comparado ao KNNImpute e ao rSVD, descritos no Capítulo 5.

O desempenho dos três algoritmos foi avaliado em testes realizados com 6 conjuntos de dados faltantes produzidos artificialmente, segundo o mecanismo MCAR, com taxas de 2%, 15%, 30%, 50%, 80% e 90%. Na primeira etapa deste experimento, os parâmetros utilizados para o SwarmBcluster foram os apresentados na Tabela 6.46. Em seguida, para as bases de 15%, 50% e 80%, as quais foram utilizadas nos experimentos de análise de influência do δ, será apresentado o ganho de desempenho do SwarmBcluster com relação ao KNNImpute e ao rSVD.

Tabela 6.50: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Jester (MCAR).

	SwarmBcluster					KNNImpute					rSVD					% Melhora	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**
2	2,908	2,721	18,550	2,079	0,00	3,239	2,613	18,398	2,594	0,00	3,361	2,602	18,022	2,777	0,00	11,37	15,56
15	2,873	2,682	19,419	2,070	0,00	3,221	2,605	19,023	2,586	0,00	3,354	2,597	19,030	2,789	0,00	12,11	16,75
30	2,892	2,655	19,073	2,125	0,00	3,264	2,635	18,289	2,629	0,00	3,374	2,606	18,940	2,815	0,00	12,85	16,65
50	2,927	2,670	19,022	2,149	0,00	3,296	2,654	19,273	2,659	0,00	3,385	2,613	18,940	2,823	0,00	12,60	15,64
80	3,083	2,788	19,501	2,284	0,00	3,397	2,731	20,284	2,748	0,00	3,466	2,702	19,370	2,867	0,00	10,20	12,42
90	3,311	3,023	23,038	2,440	0,00						3,654	2,931	19,420	2,971	0,00		10,35

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto e Min → valor mínimo do erro absoluto. A *%Melhora* indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.50 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. O algoritmo KNNImpute não foi capaz de tratar a base de dados com 90% de dados faltantes. Em todos os testes, o SwarmBcluster foi superior aos outros algoritmos. O SwarmBcluster obteve um ganho de desempenho máximo de 12,85% em relação ao KNNImpute e 16,75% em relação ao rSVD. Analisando o valor de MAE juntamente com o valor do desvio-padrão, o SwarmBcluster também obteve desempenho superior aos outros dois algoritmos. Para o valor máximo do erro absoluto, o SwarmBcluster foi inferior ao rSVD e superior ao KNNImpute nos casos com mais de 50% de dados faltantes. Os resultados para a mediana do erro absoluto se comportaram da mesma maneira que os resultados para o valor de MAE. Em todos os testes realizados nesse experimento, os três algoritmos conseguiram estimar valores idênticos aos valores reais da base de dados, como é possível observar através do valor mínimo do erro absoluto.

A Tabela 6.51 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Em todos os testes realizados, o SwarmBcluster foi superior aos outros dois algoritmos. O SwarmBcluster foi até 6,85% melhor que o KNNImpute e foi até 8,58% melhor que o rSVD. Analisando o valor de RMSE juntamente com o valor do desvio-padrão, o SwarmBcluster foi superior aos outros dois algoritmos nos casos com 30% e 50% de dados faltantes.

Tabela 6.51: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para a base Jester (MCAR).

	SwarmBcluster		**KNNImpute**		**rSVD**		**%Melhora**	
%	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**RMSE**	**D.Pad**	**1 vs. 2**	**1 vs. 3**
2	3,983	5,461	4,161	5,227	4,250	5,113	4,49	6,73
15	3,930	5,433	4,142	5,190	4,242	5,111	5,39	7,94
30	3,926	5,380	4,195	5,257	4,263	5,135	6,85	8,58
50	3,962	5,383	4,232	5,277	4,277	5,148	6,80	7,93
80	4,157	5,618	4,359	5,411	4,395	5,332	4,87	5,73
90	4,483	6,066			4,684	5,801		4,48

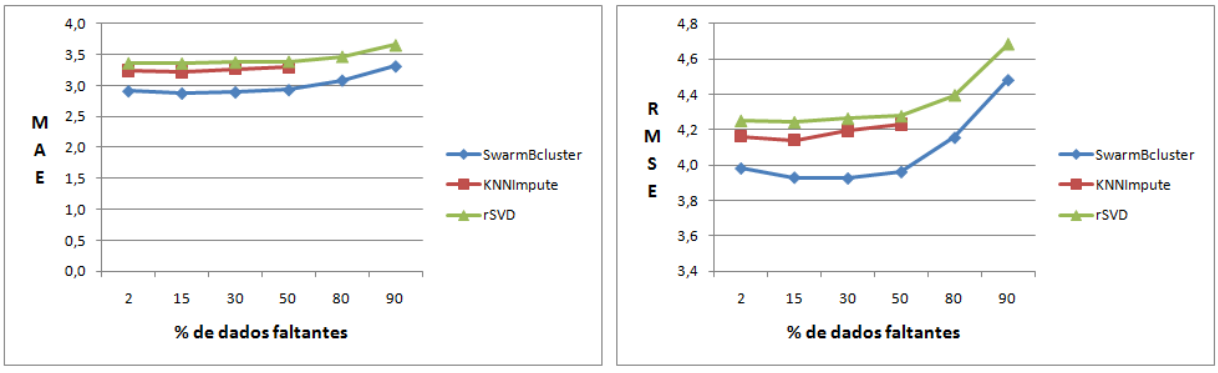
Nota: O significado da abreviação D.Pad é desvio-padrão. A *%Melhora* indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

A Tabela 6.52 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao tempo de execução para a base de dados Jester (MCAR). Como já foi comentado, o SwarmBcluster tem um custo computacional maior que os outros algoritmos.

Tabela 6.52: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao Tempo de Execução para a base Jester (MCAR).

	SwarmBcluster			**KNNImpute**	**rSVD**
%	**Tempo (min)**	**Tempo (h)**	**Tempo (d)**	**Tempo (min)**	**Tempo (min)**
2	192,92	3,22	0,13	2,97	2,19
15	1285,69	21,43	0,89	11,00	1,98
30	3903,21	65,05	2,71	19,48	1,72
50	7448,88	124,15	5,17	28,44	1,38
80	13564,44	226,07	9,42	37,95	1,02
90	19815,45	330,26	13,76		0,98

A Figura 6.25 exibe o comportamento do valor de (a) MAE e do valor de (b) RMSE. Para essa base de dados, o comportamento de MAE e RMSE é um pouco diferente. O valor de RMSE foi consideravelmente mais sensível ao aumento na quantidade de dados faltantes. O KNNImpute e o rSVD obtiveram resultados parecidos. Os resultados do SwarmBcluster foram visivelmente melhores que os resultados dos outros dois algoritmos.



(a) (b)

Figura 6.25: Comportamento do valor de (a) MAE e (b) RMSE para os algoritmos SwarmBcluster, KNNImpute e rSVD quando aplicados à base de dados Jester (MCAR).

A Figura 6.26 mostra o comportamento do valor de MAE, RMSE e do tempo de execução para o algoritmo SwarmBcluster. O tempo de execução apresentou um comportamento diferente do exibido pelo MAE e RMSE. Enquanto MAE e RMSE apresentaram um comportamento

estável, o tempo de execução cresceu consideravelmente com o aumento da quantidade de dados faltantes.

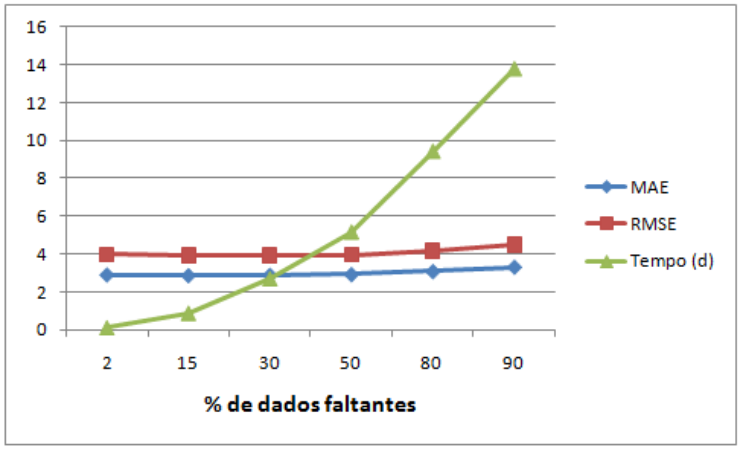


Figura 6.26: Comportamento do valor de MAE, RMSE e do Tempo de Execução para o algoritmo SwarmBcluster quando aplicado à base de dados Jester (MCAR).

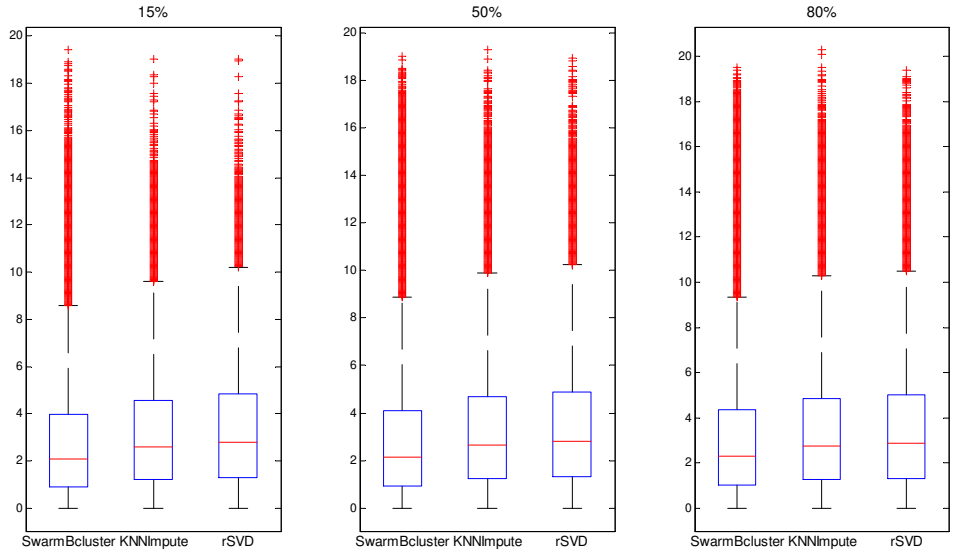


Figura 6.27: Erros absolutos produzidos por SwarmBcluster, KNNImpute e rSVD para a base de dados Jester (MCAR) com 15%, 50% e 80% de dados faltantes.

A Figura 6.27 mostra o gráfico de caixa produzido através dos erros absolutos gerados pelos algoritmos SwarmBcluster, KNNImpute e rSVD para as bases de dados com 15%, 50% e 80% de dados faltantes. Para todas essas bases de dados, o SwarmBcluster apresentou erros

absolutos mais concentrados e com menor variação que os demais algoritmos. Entretanto, todos os algoritmos apresentaram bastante outliers.

A comparação de desempenho entre o SwarmBcluster e o KNNImpute e o rSVD é exibida novamente para as bases de dados com 15%, 50% e 80% de dados faltantes, as quais participaram da análise de sensibilidade na variação do parâmetro δ. Para as bases de dados com 15% e 50% de dados faltantes, o melhor valor de δ foi 2. Para a base de dados com 80% de dados faltantes, o melhor valor de δ foi 4. Os outros parâmetros utilizados foram iguais àqueles apresentados na Tabela 6.46.

A Tabela 6.53 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE. Para a base de dados com 15% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 12,11% para 14,31% e com relação ao rSVD houve um aumento de 16,75% para 19,04%. Para a base de dados com 50% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 12,6% para 15% e com relação ao rSVD aumentou de 15,64% para 18,12%. Para a base de dados com 80% de dados faltantes, o ganho no valor de MAE com relação ao KNNImpute aumentou de 10,2% para 12,13% e com relação ao rSVD aumentou de 12,42% para 14,39%. Os valores do desvio-padrão e da mediana também são menores neste teste do que os obtidos com $\delta = 6$. Com relação ao valor máximo do erro absoluto, este teste foi melhor para os casos com 15% e 80% de dados faltantes.

A Tabela 6.54 traz a comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE. Para a base de dados com 15% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 5,39% para 7,51% e com relação ao rSVD aumentou de 7,94% para 10,1%. Para a base de dados com 50% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 6,8% para 9,2% e com relação ao rSVD aumentou de 7,93% para 10,37%. Para a base de dados com 80% de dados faltantes, o ganho no valor de RMSE com relação ao KNNImpute aumentou de 4,87% para 6,74% e com relação ao rSVD aumentou de 5,73% para 7,61%. Para todas as bases de dados, o valor do desvio-padrão é menor neste teste do que o obtido com $\delta = 6$.

Tabela 6.53: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de MAE para a base Jester (MCAR) utilizada nos experimentos de sensibilidade paramétrica.

	SwarmBcluster					KNNImpute					rSVD					%MelAnter		%MelAtual	
%	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**MAE**	**D.Pad**	**Max**	**Med**	**Min**	**1 vs. 2**	**1 vs. 3**	**1 vs. 2**	**1 vs. 3**
15	2,818	2,628	18,806	2,017	0,00	3,221	2,605	19,023	2,586	0,00	3,354	2,597	19,030	2,789	0,00	12,11	16,75	14,31	19,04
50	2,866	2,608	19,630	2,121	0,00	3,296	2,654	19,273	2,659	0,00	3,385	2,613	18,940	2,823	0,00	12,60	15,64	15,00	18,12
80	3,030	2,738	19,420	2,246	0,00	3,397	2,731	20,284	2,748	0,00	3,466	2,702	19,370	2,867	0,00	10,20	12,42	12,13	14,39

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, Max → valor máximo do erro absoluto, Med → mediana do erro absoluto, Min → valor mínimo do erro absoluto, %MelAnter → porcentagem de melhora anterior e %MelAtual → porcentagem de melhora atual. A porcentagem de melhora indica, percentualmente, o quanto o valor de MAE obtido pelo SwarmBcluster é melhor que o valor de MAE obtido pelo KNNImpute e rSVD, respectivamente.

Tabela 6.54: Comparação entre os algoritmos SwarmBcluster, KNNImpute e rSVD com relação ao valor de RMSE para as bases Jester (MCAR) utilizadas nos experimentos de sensibilidade paramétrica.

	SwarmBcluster		KNNImpute		rSVD		%MelAnter		%MelAtual	
%	RMSE	D.Pad	RMSE	D.Pad	RMSE	D.Pad	1 vs. 2	1 vs. 3	1 vs. 2	1 vs. 3
15	3,853	5,248	4,142	5,190	4,242	5,111	5,39	7,94	7,51	10,10
50	3,875	5,244	4,232	5,277	4,277	5,148	6,80	7,93	9,20	10,37
80	4,084	5,527	4,359	5,411	4,395	5,332	4,87	5,73	6,74	7,61

Nota: Os significados das abreviações são: D.Pad → desvio-padrão, %MelAnter → porcentagem de melhora anterior e %MelAtual → % melhora de atual. A porcentagem de melhora indica, percentualmente, o quanto o valor de RMSE obtido pelo SwarmBcluster é melhor que o valor de RMSE obtido pelo KNNImpute e rSVD, respectivamente.

6.4 Discussão

A análise de sensibilidade paramétrica mostrou que a escolha apropriada dos parâmetros pode melhorar consideravelmente o desempenho do algoritmo.

Dentre os parâmetros testados, δ, max_it e n_ants, aquele que exerceu maior influência na qualidade da imputação foi o δ, que controla diretamente a intensidade máxima de ruído presente no bicluster. Percebeu-se que o melhor valor de δ depende da quantidade de dados faltantes. Para quantidades pequenas ou médias, o melhor valor de δ tende a ser menor que para grandes quantidades de dados faltantes, isso porque quando a base contém muitos dados faltantes a intensidade do ruído, por conta da ausência de valores, é maior e, consequentemente, deve-se permitir que os biclusters encontrados tenham um ruído também maior. Para definir um valor de δ, é necessário conhecer o intervalo em que excursionam os valores da base de dados. Percebeu-se que o melhor valor de δ encontrado foi sempre menor que o limite máximo desse intervalo. Para a base de dados Yeast, os valores estão no intervalo [0, 595] e os melhores valores de δ foram 200 e 400. Para a base de dados Human, os valores estão no intervalo [-605, 640]. Colocando isso numa escala não-negativa (os valores de δ não podem ser negativos), tem-se [0, 1245] e os melhores valores de δ foram 600 e 1200. Para a base de dados Jester, os valores estão no intervalo [0,05; 19,47] e os melhores valores de δ foram 2 e 4. Para todas as bases de dados utilizadas (Yeast, Human e Jester), é interessante notar que o melhor δ encontrado para o caso com 80% de dados faltantes foi o dobro do melhor δ encontrado para os casos com 15% e 50% de dados faltantes.

Além de influenciar a qualidade da imputação, o valor de δ também influenciou significativamente o tempo de execução do programa. Na maior parte dos casos, esse parâmetro obteve uma correlação negativa com o custo computacional, pois quanto menor seu valor a tendência é encontrar biclusters menores e, consequentemente, será necessário encontrar mais biclusters para cobrir todos os valores faltantes. Apesar disso, em alguns casos ocorreu o oposto, mostrando que essa relação pode apresentar particularidades dependendo da base de dados e da quantidade de dados faltantes.

O segundo parâmetro a exercer maior influência na qualidade das imputações foi max_it, mais notavelmente para os casos com maior número de dados faltantes. Isso ocorre pelo mesmo motivo que um δ maior também melhorou os resultados nesses casos: o parâmetro max_it influencia em biclusters com maiores volumes. Um maior volume em bicluster nesses casos faz com que a porcentagem de dados faltantes dentro do bicluster seja menor, reduzindo a influência do ruído e tornando o problema de otimização correspondente factível.

O parâmetro n_ants foi o que menos influenciou a qualidade da imputação. Quanto maior o número de formigas, maior o número de soluções (biclusters) geradas a cada iteração do ACO. Uma possibilidade para a pouca influência é que a qualidade das soluções encontradas por cada formiga é parecida e um número alto delas exerce pouca influência na atualização do feromônio. Em DE FRANÇA (2010), os resultados obtidos pelo SwarmBcluster apenas na tarefa de biclusterização mostraram que o aumento dos parâmetros max_it e n_ants influenciou apenas o volume médio dos biclusters, enquanto o resíduo médio sofreu um leve aumento. Isso pode significar que, para o problema de imputação, o volume do bicluster pode não ter tanta influência na qualidade de imputação quanto o MSR tem.

O desempenho da heurística construtiva também foi avaliado quanto à qualidade da imputação obtida. Os resultados mostraram que, com um custo computacional bastante reduzido, foi possível obter uma qualidade de imputação próxima ao experimento que analisou a influência do valor de δ.

Os experimentos de comparação de desempenho para imputação única entre o SwarmBcluster, o KNNImpute e o rSVD mostraram que, na grande maioria das vezes, o desempenho do SwarmBcluster foi consideravelmente melhor. A única desvantagem desse algoritmo foi o custo computacional. Como já foi dito, a razão para este alto custo é que o SwarmBcluster cria vários modelos locais da base de dados para encontrar o subconjunto de

objetos e atributos mais adequados para imputar um subconjunto de dados faltantes. Já os algoritmos KNNImpute e rSVD criam apenas um único modelo global para toda a base de dados. Mas, considerando a qualidade das imputações, o SwarmBcluster pode ser uma boa escolha para aplicações off-line. Além do mais, ao contrário do KNNImpute, o SwarmBcluster conseguiu tratar todas as bases de dados com altas porcentagens de dados faltantes, as quais, obviamente, são mais difíceis de tratar.

Dentre os três mecanismos de dados faltantes, o SwarmBcluster não se destacou tanto no caso do mecanismo MNAR. Ao contrário do que ocorreu nos experimentos realizados com os mecanismos MCAR e MAR, com o MNAR os resultados obtidos pelo SwarmBcluster não foram consideravelmente melhores que os resultados obtidos pelo: (*i*) rSVD nos experimentos realizados com a base de dados Yeast; e (*ii*) KNNImpute nos experimentos realizados com a base de dados Human.

6.5 Síntese do Capítulo

Neste capítulo, foram apresentadas e examinadas: (*i*) a análise de sensibilidade paramétrica do algoritmo SwarmBcluster; e (*ii*) as comparações de desempenho para imputação única entre os algoritmos SwarmBcluster, KNNImpute e rSVD.

Os experimentos mostraram que apenas o uso da heurística construtiva do SwarmBcluster foi suficiente para obter bons resultados de imputação frente a outros métodos estudados e que o ajuste correto do parâmetro δ para cada base de dados pode melhorar significativamente o desempenho geral do algoritmo.

O próximo capítulo apresenta os experimentos realizados com biclusterização e MI. Para comparar o desempenho do SwarmBcluster, será utilizado o algoritmo NORM (SCHAFER, 1999), o qual é um programa clássico de MI. O KNNImpute e o rSVD não serão utilizados neste próximo capítulo, pois são algoritmos determinísticos. O princípio da MI é realizar imputações que possuam variação aleatória, de modo que os valores sejam únicos entre cada conjunto imputado, mas compartilhem uma relação comum subjacente aos dados. Essa variação aleatória pode eliminar o viés que é endêmico às imputações determinísticas. Logicamente, é possível variar os parâmetros do KNNImpute e do rSVD para que eles gerem imputações diferentes para cada dado faltante, o que também poderia ser feito com o SwarmBcluster. Em princípio, esse não

é o procedimento comum para os métodos de MI e não é o foco de estudo deste trabalho. Mas, certamente, esse estudo é interessante e pode ser realizado em trabalhos futuros.

Capítulo 7

Biclusterização e imputação múltipla

Este capítulo apresenta os experimentos realizados utilizando a biclusterização como modelo para a técnica de MI. Para mostrar a eficácia da biclusterização, os resultados obtidos foram comparados aos produzidos pelo programa NORM (SCHAFER, 1999). Esse programa foi escolhido por ser gratuito, amigável ao usuário e bastante conhecido, ele também foi utilizado nos experimentos realizados com MI por MCKNIGHT *et al.* (2007).

Como já foi mencionado na Subseção 3.6.3, os principais procedimentos do NORM são: um algoritmo de Esperança e Maximização (EM) e um procedimento de *Data Augmentation* (DA). O algoritmo EM pode ser utilizado não só para obter as estimativas iniciais dos parâmetros, mas também para estimar o número de iterações necessárias do procedimento DA. SCHAFER (1999) afirma que esse valor deve ser igual ou superior ao número de iterações necessárias para o algoritmo EM convergir. Por isso, o número de iterações do DA foi definido com base no número de iterações necessárias para o EM convergir mais uma pequena folga. Em todos os experimentos realizados com o NORM, o critério de convergência do EM foi 10^{-4}.

Os experimentos de MI foram realizados com os conjuntos de dados artificiais criados a partir das bases de dados Yeast e Human, os quais foram apresentados no Capítulo 6. Para o mecanismo MCAR, foram escolhidos os conjuntos de dados com 15%, 50% e 80% de dados faltantes, os quais também foram utilizados nos experimentos de análise de sensibilidade paramétrica. Para os mecanismos MAR e MNAR, foram escolhidos os conjuntos de dados com 40% e 80% de dados faltantes, ou seja, um conjunto de dados com uma quantidade razoável de dados faltantes e um conjunto de dados com uma grande quantidade de dados faltantes.

A normalidade das bases de dados Yeast e Human foi verificada através dos testes Jarque-Bera (JARQUE & BERA, 1987), Lilliefors (LILLIEFORS, 1967) e Kolmogorov-Smirnov (MASSEY, 1951) ao nível de significância de 5%. Para ambas, todos os testes mostraram que não há evidências suficientes para afirmar que elas possuem distribuição normal. Também se tentou verificar a normalidade dessas bases de dados graficamente, através do *gráfico de probabilidade*

normal. Para cada atributo, esse gráfico mostra os dados com o símbolo '+' e juntamente uma reta sobreposta que une o primeiro ao terceiro quartil. Essa reta é extrapolada até os confins da amostra para ajudar a avaliar a linearidade dos dados. Se os dados de um atributo são normalmente distribuídos, o gráfico tende a ser linear. Outros tipos de distribuição irão introduzir curvatura no gráfico. A Figura 7.1 mostra os gráficos de probabilidade normal construídos a partir dos 17 atributos da base de dados Yeast e dos 40 atributos da base de dados Human. Embora não se possa afirmar que essas duas bases de dados possuem distribuição normal a partir desses gráficos, é possível perceber que considerar tal distribuição não é uma suposição muito forte. Principalmente para os atributos da base de dados Yeast, é possível visualizar que a maior parte dos dados de cada um dos atributos se ajusta bem a uma reta. Para a base de dados Human, observa-se que apesar de existirem duas regiões em que os dados apresentam curvatura, a maior parte desses dados se ajusta a uma reta.

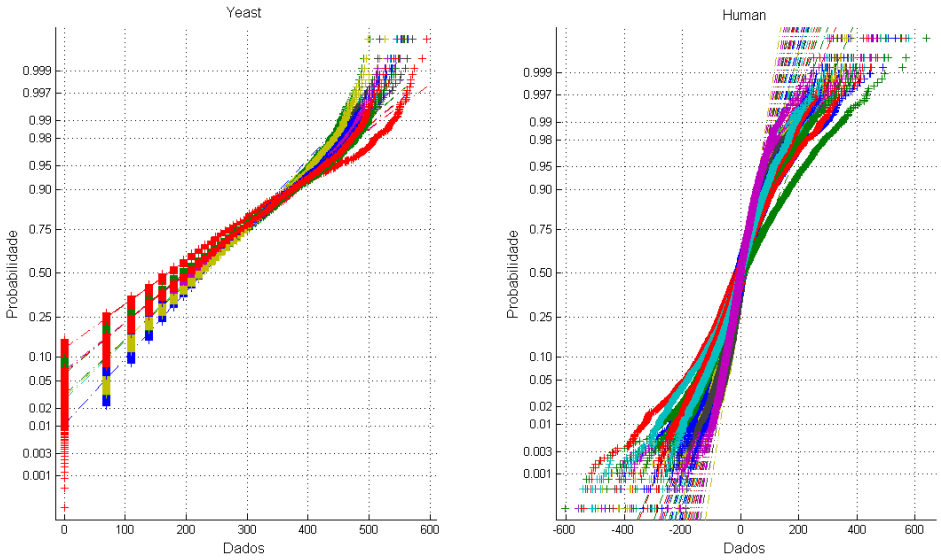


Figura 7.1: Gráfico de probabilidade normal das bases de dados Yeast e Human.

Para ser possível utilizar as métricas definidas por RUBIN (1987) (apresentadas na Subseção 3.6.1.3) é necessário definir um parâmetro de interesse. O parâmetro de interesse escolhido para este trabalho é a média aritmética. Ela foi escolhida por se tratar de uma medida amplamente conhecida e de fácil entendimento. Logo, para cada um dos $x \geq 2$ conjuntos de dados completos produzidos a partir do processo de imputação, a média aritmética de cada atributo foi calculada. Também foi calculado o erro padrão da média. Assim, os resultados podem ser agregados, como

descrito na Subseção 3.6.1.3, e a taxa de informação faltante (γ) pode ser calculada, como descrito na Subseção 3.6.1.4. Os resultados dos experimentos de MI exibirão para cada atributo do conjunto de dados: a média global estimada, o intervalo de confiança de 95% da média global estimada, a variabilidade dos erros padrão calculados ($\overline{U}$), a variância das médias aritméticas estimadas (B), a variância total (T), os graus de liberdade (df), o aumento relativo na variância devido aos dados faltantes (r), a taxa de informação faltante (γ) e a eficiência das estimativas calculadas com $x \geq 2$ imputações em relação a uma baseada em um número infinito de imputações (*Efic*.).

Na maioria dos testes, o número de imputações realizadas foi $x = 5$, ou seja, para cada dado faltante 5 valores foram estimados, produzindo 5 conjuntos de dados completos. Esse valor foi escolhido porque está dentro do intervalo [3, 10] estabelecido por RUBIN (1987), tendo sido utilizado nos experimentos realizados com MI por MCKNIGHT *et al.* (2007).

Tanto o algoritmo SwarmBcluster como o algoritmo DA utilizado pelo NORM são estocásticos. Logo, a execução desses algoritmos x vezes produz x imputações aleatórias para cada dado faltante. A cada execução desses algoritmos, um conjunto de dados completo é produzido. As bases de dados completas produzidas pelo NORM e pelo SwarmBcluster serão analisadas individualmente e, em seguida, os resultados produzidos serão agregados e a taxa de informação faltante calculada. A Figura 7.2 mostra: (*i*) em que passo do processo de MI o SwarmBcluster e o NORM foram utilizados; e (*ii*) que os passos de *análise*, *agregação dos resultados* e *cálculo da informação faltante* foram implementados conforme descrição apresentada nas Subseções 3.6.1.2 a 3.6.1.4.

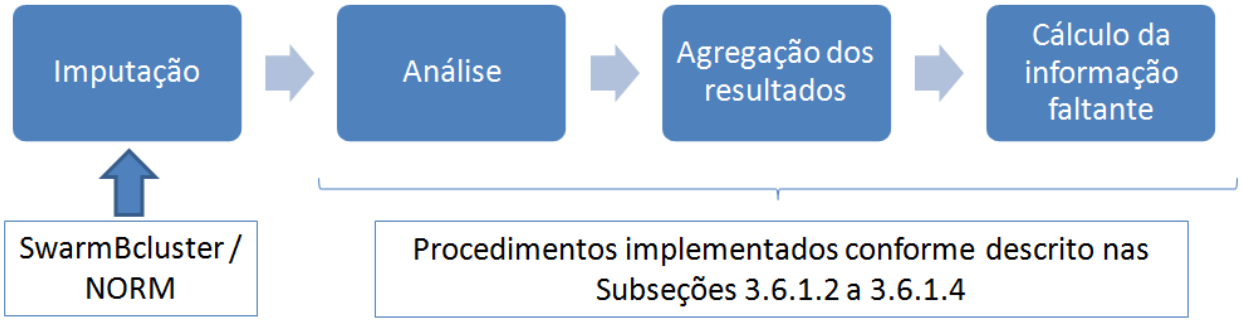


Figura 7.2: Descrição da técnica de MI utilizada.

Também será mostrado o quanto se pode ganhar na qualidade da imputação quando um dado faltante é estimado com base em imputações múltiplas em vez de uma imputação única (IU). Isso será feito de duas maneiras. A primeira é utilizando a média dos *x* valores estimados para cada dado faltante e a segunda é utilizando a mediana dos *x* valores estimados para cada dado faltante. Em trabalhos futuros, pretende-se investigar outras maneiras de agrupar os *x* valores estimados em um único valor para ser imputado. Novamente, as métricas utilizadas para medir a qualidade da imputação serão MAE e RMSE.

7.1 Base de dados Yeast

Os experimentos listados a seguir foram realizados com a base de dados Yeast. Como o parâmetro de interesse escolhido para este trabalho é a média aritmética, a Tabela 7.1 exibe a média dos 17 atributos da base de dados Yeast.

Tabela 7.1: Média aritmética dos 17 atributos da base de dados Yeast.

Atributo	1	2	3	4	5	6	7	8	9
Média	227,9702	218,2613	208,1620	217,0035	211,1506	221,2280	214,4032	214,1298	222,7113
Atributo	10	11	12	13	14	15	16	17	
Média	185,6461	208,8088	214,6929	223,5406	209,1877	210,3227	202,6409	209,0232	

A primeira parte desta seção traz os experimentos realizados com as bases de dados com 15%, 50% e 80% de dados faltantes segundo o mecanismo MCAR. A segunda parte traz os experimentos realizados com as bases de dados com 40% e 80% de dados faltantes segundo o mecanismo MAR. A terceira parte traz os experimentos realizados com as bases de dados com 40% e 80% de dados faltantes segundo o mecanismo MNAR.

7.1.1 Dados faltantes segundo o mecanismo MCAR

As bases de dados com 15%, 50% e 80% de dados faltantes, segundo o mecanismo MCAR, utilizadas nas Subseções 6.1.1 a 6.1.5 foram utilizadas neste experimento.

Tabela 7.2: Resultados da MI utilizando biclusterização – Yeast com 15% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	228,1484	224,4236	231,8732	3,6066	0,0041	3,6116	2142516,2429	0,0014	0,0014	0,9997
2	218,8230	214,6769	222,9690	4,4667	0,0066	4,4746	1263029,3788	0,0018	0,0018	0,9996
3	208,3807	203,9121	212,8494	5,1967	0,0012	5,1981	56601306,2073	0,0003	0,0003	0,9999
4	216,7620	212,6561	220,8680	4,3873	0,0010	4,3885	53110269,2527	0,0003	0,0003	0,9999
5	211,3846	207,0538	215,7153	4,8809	0,0011	4,8822	59708822,4231	0,0003	0,0003	0,9999
6	221,2174	217,2511	225,1838	4,0906	0,0038	4,0951	3264634,8166	0,0011	0,0011	0,9998
7	214,4304	210,2570	218,6037	4,5329	0,0007	4,5337	121920289,0703	0,0002	0,0002	1,0000
8	214,1625	210,0181	218,3070	4,4647	0,0054	4,4712	1922390,8271	0,0014	0,0014	0,9997
9	222,4900	218,6088	226,3712	3,9164	0,0040	3,9212	2707211,5139	0,0012	0,0012	0,9998
10	185,9423	181,0086	190,8761	6,3360	0,0004	6,3364	828336947,8423	0,0001	0,0001	1,0000
11	208,9721	204,5324	213,4118	5,1264	0,0038	5,1309	5080338,5220	0,0009	0,0009	0,9998
12	214,6243	210,3403	218,9083	4,7765	0,0006	4,7773	164537525,3943	0,0002	0,0002	1,0000
13	223,4094	219,5211	227,2976	3,9347	0,0007	3,9355	98203671,3355	0,0002	0,0002	1,0000
14	209,3024	204,9344	213,6704	4,9646	0,0016	4,9665	26153859,4551	0,0004	0,0004	0,9999
15	210,5042	206,2067	214,8017	4,8041	0,0029	4,8075	7863328,8928	0,0007	0,0007	0,9999
16	202,9409	198,3558	207,5259	5,4692	0,0026	5,4723	12479438,3659	0,0006	0,0006	0,9999
17	208,9951	204,5994	213,3909	5,0286	0,0010	5,0298	66086031,8305	0,0002	0,0002	1,0000

Tabela 7.3: Resultados da MI utilizando NORM – Yeast com 15% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	228,0652	224,2387	231,8917	3,6475	0,1366	3,8114	2163,2862	0,0449	0,0439	0,9913
2	218,5897	214,4327	222,7466	4,4713	0,0224	4,4982	111922,7329	0,0060	0,0060	0,9988
3	207,9962	203,4660	212,5263	5,2487	0,0779	5,3421	13077,6492	0,0178	0,0176	0,9965
4	216,8736	212,7476	220,9995	4,4196	0,0098	4,4314	562255,9246	0,0027	0,0027	0,9995
5	211,3104	206,9423	215,6785	4,9237	0,0359	4,9668	53131,4626	0,0088	0,0087	0,9983
6	221,1552	217,1743	225,1362	4,0929	0,0271	4,1254	64485,6486	0,0079	0,0079	0,9984
7	214,2697	210,0311	218,5083	4,5775	0,0825	4,6766	8918,6447	0,0216	0,0214	0,9957
8	213,8866	209,6953	218,0779	4,5033	0,0580	4,5729	17267,0252	0,0155	0,0153	0,9969
9	222,3754	218,4855	226,2654	3,9117	0,0227	3,9389	83493,0770	0,0070	0,0069	0,9986
10	185,3058	180,3544	190,2573	6,3608	0,0176	6,3819	367252,9673	0,0033	0,0033	0,9993
11	208,5931	204,1324	213,0538	5,1561	0,0195	5,1796	195273,1947	0,0045	0,0045	0,9991
12	214,9646	210,6570	219,2723	4,8059	0,0203	4,8303	157112,8434	0,0051	0,0051	0,9990
13	223,6954	219,8042	227,5865	3,9271	0,0118	3,9413	308087,3046	0,0036	0,0036	0,9993
14	209,1037	204,6886	213,5189	5,0142	0,0502	5,0744	28353,0342	0,0120	0,0119	0,9976
15	210,2602	205,9407	214,5796	4,8372	0,0162	4,8567	248935,7537	0,0040	0,0040	0,9992
16	202,8643	198,2522	207,4763	5,5203	0,0140	5,5371	432502,9766	0,0031	0,0030	0,9994
17	208,9212	204,4867	213,3556	5,0717	0,0393	5,1189	47201,9615	0,0093	0,0092	0,9982

Devido ao melhor tempo de execução e à qualidade dos resultados obtidos pela heurística na Subseção 6.1.4, o SwarmBcluster foi parametrizado como naquele experimento, com $\delta = 200$

para as bases com 15% e 50% de dados faltantes, e $\delta = 400$ para a base com 80% de dados faltantes.

As Tabelas 7.2 e 7.3 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 15% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 17. Logo, o número de iterações de cada ciclo do DA foi igual a 20, totalizando 100 iterações para produzir os cinco conjuntos de dados completos. Apesar de o NORM ter obtido bons resultados para *df*, os resultados produzidos pela técnica de biclusterização foram superiores, indicando que a variância total foi bem estimada. A biclusterização também obteve resultados melhores para *r*, o que indica maior estabilidade nos parâmetros estimados, consequentemente refletindo em uma taxa menor de informação faltante (γ). A biclusterização também foi superior quanto à eficiência das estimativas obtidas com 5 imputações quando comparadas a um número infinito de imputações (*Efic.*), o que indica que pouco se ganharia aumentando o número de imputações. Tanto para o NORM quanto para a biclusterização, a média real dos atributos da base de dados Yeast pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.4: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 15% de dados faltantes MCAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
15,814	22,017	15,867	22,118	22,116	29,283	22,937	30,228	16,278	22,712

A Tabela 7.4 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Yeast com 15% de dados faltantes MCAR. Além de o SwarmBcluster ter produzido resultados melhores que os do NORM, tanto a imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que a imputação baseada em uma única execução da heurística. Obviamente, realizar imputações múltiplas para um dado faltante, no lugar de uma única, tem um custo computacional maior. Portanto, o ganho na qualidade das imputações deveria justificar o maior custo computacional, o que não acontece nesse caso, pois o ganho no valor de MAE é de apenas 2,85% e o ganho no valor de RMSE é de apenas 3,06%. Entretanto, é importante salientar que as *x* execuções do algoritmo SwarmBcluster

são independentes, podendo, portanto, serem realizadas simultaneamente. Desse modo, não haveria impacto no tempo de espera pelo resultado.

As Tabelas 7.5 e 7.6 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 50% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 80. Logo, o número de iterações de cada ciclo do DA foi igual a 90, totalizando 450 iterações para produzir os cinco conjuntos de dados completos. Os resultados deste teste foram parecidos com o caso de 15% de dados faltantes. A biclusterização também obteve melhores resultados para *r*, refletindo em uma taxa menor de informação faltante (γ). No caso da biclusterização, a *Eficiência* foi superior a 99% para todos os atributos. Já para o NORM, houve um atributo com *Eficiência* inferior a 95%. Tanto para o NORM quanto para a biclusterização, a média real dos atributos da base de dados Yeast pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.5: Resultados da MI utilizando biclusterização – Yeast com 50% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	227,7611	224,0553	231,4669	3,5243	0,0421	3,5748	20036,2851	0,0143	0,0142	0,9972
2	217,7119	213,6104	221,8135	4,3626	0,0138	4,3791	281684,0632	0,0038	0,0038	0,9992
3	207,3684	202,9533	211,7835	5,0560	0,0152	5,0742	310484,9235	0,0036	0,0036	0,9993
4	216,4219	212,3667	220,4770	4,2467	0,0282	4,2805	64218,6770	0,0080	0,0079	0,9984
5	210,2003	205,8894	214,5113	4,7453	0,0770	4,8377	10973,6368	0,0195	0,0193	0,9962
6	219,4769	215,5363	223,4175	4,0087	0,0279	4,0422	58279,7943	0,0084	0,0083	0,9983
7	214,1200	209,9885	218,2515	4,3976	0,0382	4,4434	37620,7300	0,0104	0,0104	0,9979
8	214,1277	209,9994	218,2560	4,3514	0,0708	4,4364	10894,8534	0,0195	0,0193	0,9961
9	220,9681	217,0712	224,8650	3,9223	0,0256	3,9530	66269,0587	0,0078	0,0078	0,9984
10	185,6793	180,8020	190,5566	6,1783	0,0116	6,1922	796901,2828	0,0022	0,0022	0,9996
11	207,6501	203,2231	212,0772	5,0160	0,0714	5,1017	14174,5105	0,0171	0,0169	0,9966
12	214,0013	209,7387	218,2638	4,6622	0,0563	4,7297	19635,7258	0,0145	0,0144	0,9971
13	221,5954	217,6997	225,4911	3,9404	0,0085	3,9506	602789,5394	0,0026	0,0026	0,9995
14	208,8402	204,5309	213,1495	4,7921	0,0348	4,8339	53600,9165	0,0087	0,0087	0,9983
15	209,8779	205,5973	214,1586	4,7186	0,0428	4,7699	34553,2713	0,0109	0,0108	0,9978
16	202,0752	197,5685	206,5819	5,2738	0,0109	5,2869	647861,7341	0,0025	0,0025	0,9995
17	207,6192	203,2562	211,9822	4,9333	0,0183	4,9552	203802,1980	0,0044	0,0044	0,9991

A Tabela 7.7 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Yeast com 50% de dados faltantes MCAR. O SwarmBcluster foi consideravelmente melhor que o NORM nas imputações realizadas com base nas MI. Mais uma vez, tanto a

imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que a imputação baseada em uma única execução da heurística. Os ganhos em termos de qualidade com o emprego de MI foram maiores que os obtidos no teste com a base de dados com 15% de dados faltantes. O ganho no valor de MAE foi de 6,49% e o ganho no valor de RMSE foi de 8,83%.

Tabela 7.6: Resultados da MI utilizando NORM – Yeast com 50% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	228,6797	224,3056	233,0539	3,6394	1,1176	4,9805	55,1693	0,3685	0,2944	0,9444
2	219,1197	214,7387	223,5007	4,5082	0,4065	4,9961	419,5264	0,1082	0,1019	0,9800
3	209,4096	204,4823	214,3369	5,2571	0,8856	6,3198	141,4695	0,2021	0,1797	0,9653
4	216,9638	212,8351	221,0924	4,3417	0,0795	4,4371	8645,5858	0,0220	0,0217	0,9957
5	211,5611	206,9268	216,1954	4,9154	0,5627	5,5906	274,2239	0,1374	0,1271	0,9752
6	220,5276	216,1787	224,8766	4,0763	0,7058	4,9233	135,1545	0,2078	0,1840	0,9645
7	214,5038	210,1374	218,8703	4,5345	0,3571	4,9630	536,5749	0,0945	0,0897	0,9824
8	214,5990	210,3522	218,8457	4,5304	0,1369	4,6947	3266,9971	0,0363	0,0356	0,9929
9	222,9188	218,9288	226,9088	3,9052	0,1991	4,1441	1203,2358	0,0612	0,0592	0,9883
10	185,8847	180,9163	190,8530	6,3268	0,0823	6,4256	16947,0844	0,0156	0,0155	0,9969
11	209,0006	204,2773	213,7238	5,2234	0,4865	5,8072	395,7808	0,1118	0,1050	0,9794
12	215,2507	210,9024	219,5990	4,7886	0,1110	4,9218	5456,8516	0,0278	0,0274	0,9945
13	223,3080	219,3720	227,2441	3,9325	0,0836	4,0329	6466,5136	0,0255	0,0252	0,9950
14	208,8650	204,4088	213,3213	4,9178	0,2095	5,1692	1690,8203	0,0511	0,0498	0,9901
15	211,3847	206,9945	215,7748	4,8330	0,1534	5,0170	2971,7720	0,0381	0,0373	0,9926
16	202,3867	197,6122	207,1611	5,4189	0,4291	5,9339	531,1051	0,0950	0,0902	0,9823
17	208,9482	204,4263	213,4702	5,0576	0,2209	5,3227	1612,6645	0,0524	0,0510	0,9899

Tabela 7.7: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 50% de dados faltantes MCAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
17,460	24,563	17,197	24,244	25,005	33,242	26,047	34,486	18,392	26,592

As Tabelas 7.8 e 7.9 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 80% de dados faltantes. O número de imputações foi igual a 3 porque foi o máximo de imputações que o NORM conseguiu realizar. Depois da terceira imputação, o NORM retornou um erro devido à matriz de covariância não ser definida positiva.

O algoritmo EM convergiu na iteração 1109. Logo, o número de iterações de cada ciclo do DA foi igual a 1220, totalizando 3660 iterações para produzir os três conjuntos de dados completos. Tanto a biclusterização como o NORM produziram resultados significativamente piores em relação aos dois primeiros testes. Para vários atributos, a informação faltante estimada (γ) foi maior que 30%. Ao contrário dos testes anteriores, a biclusterização não foi superior ao NORM para todos os atributos. Além disso, apenas para o NORM a média real dos atributos da base de dados Yeast pertence ao intervalo estabelecido por uma confiança de 95%. Lembrando que os dados dos atributos da base de dados Yeast se ajustam relativamente bem à reta no gráfico de probabilidade normal (Figura 7.1), esse êxito do NORM pode estar no fato de que ele foi construído para estimar parâmetros de interesse sob uma distribuição normal multivariada. Enquanto isso, o SwarmBcluster objetiva apenas estimar o valor faltante, sem considerar nenhuma distribuição para os dados. A grande vantagem da biclusterização, neste caso, é que, ao contrário do NORM, ela pode trabalhar com um número maior de imputações.

Tabela 7.8: Resultados da MI utilizando biclusterização – Yeast com 80% de dados faltantes MCAR (3 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*Df*	*r*	γ	*Efic.*
1	212,9364	206,4525	219,4202	3,6247	5,4890	10,9433	4,4716	2,0191	0,7574	0,7984
2	204,6345	198,2824	210,9866	4,2026	4,7254	10,5032	5,5579	1,4992	0,6934	0,8123
3	193,9531	189,0933	198,8129	4,8886	0,9445	6,1479	47,6660	0,2576	0,2362	0,9270
4	203,4883	198,9145	208,0620	4,3534	0,8191	5,4456	49,7230	0,2509	0,2309	0,9285
5	201,5348	196,9436	206,1260	4,5380	0,7118	5,4871	66,8480	0,2091	0,1967	0,9385
6	206,5643	202,1851	210,9435	4,0887	0,6774	4,9920	61,0881	0,2209	0,2065	0,9356
7	204,0106	198,4994	209,5218	4,4089	2,6232	7,9064	10,2203	0,7933	0,5267	0,8506
8	202,5645	197,9710	207,1580	4,3551	0,8531	5,4925	46,6341	0,2612	0,2390	0,9262
9	207,6610	202,6112	212,7107	4,0305	1,9555	6,6379	12,9622	0,6469	0,4689	0,8648
10	171,0239	165,1007	176,9471	5,7463	2,5398	9,1327	14,5464	0,5893	0,4425	0,8715
11	196,0002	191,3157	200,6847	4,8553	0,6428	5,7123	88,8570	0,1765	0,1685	0,9468
12	202,2584	197,9402	206,5766	4,6232	0,1731	4,8539	884,9243	0,0499	0,0497	0,9837
13	209,7786	204,5677	214,9896	4,1219	2,2098	7,0684	11,5098	0,7148	0,4972	0,8578
14	198,1616	193,6293	202,6940	4,6533	0,5206	5,3474	118,7131	0,1492	0,1441	0,9542
15	198,5520	194,2391	202,8649	4,6704	0,1286	4,8420	1594,0489	0,0367	0,0366	0,9879
16	192,4025	187,8353	196,9696	5,0190	0,3080	5,4297	349,5647	0,0818	0,0809	0,9737
17	195,2998	189,6620	200,9375	4,8194	2,5907	8,2736	11,4739	0,7167	0,4980	0,8576

A Tabela 7.10 exibe os resultados da estimação dos dados faltantes utilizando MI com 3 imputações para a base de dados Yeast com 80% de dados faltantes MCAR. O SwarmBcluster produziu resultados melhores que o NORM para MAE, mas não para RMSE. Como o RMSE

enfatiza os erros maiores, esse resultado significa que o SwarmBcluster produziu erros maiores que o NORM, mas em média produziu imputações melhores que o NORM. Além disso, mais uma vez tanto a imputação pela média como pela mediana das 3 imputações de cada dado faltante produziram resultados melhores que a heurística executada uma única vez. O ganho no valor de MAE foi de 5,57% e o ganho no valor de RMSE foi de 8,27%.

Tabela 7.9: Resultados da MI utilizando NORM – Yeast com 80% de dados faltantes MCAR (3 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	225,8953	221,2202	230,5705	3,3715	1,7386	5,6896	12,0488	0,6875	0,4862	0,8605
2	218,2819	213,9445	222,6192	4,4220	0,3564	4,8971	212,4582	0,1074	0,1054	0,9661
3	203,7737	198,8410	208,7063	5,4690	0,6484	6,3336	107,3314	0,1581	0,1522	0,9517
4	218,1541	213,4332	222,8750	4,3198	1,1112	5,8015	30,6629	0,3430	0,2996	0,9092
5	212,8780	206,6538	219,1022	4,9596	3,8437	10,0845	7,7439	1,0333	0,5997	0,8334
6	220,2975	215,9927	224,6023	3,9397	0,6631	4,8238	59,5382	0,2244	0,2094	0,9348
7	215,0862	209,1336	221,0387	4,9146	3,2317	9,2235	9,1642	0,8767	0,5548	0,8439
8	213,7732	208,6506	218,8958	4,6753	1,6166	6,8307	20,0863	0,4610	0,3748	0,8889
9	221,6869	217,1912	226,1826	3,9610	0,9752	5,2613	32,7437	0,3283	0,2893	0,9121
10	183,6245	175,4370	191,8119	6,7907	7,9941	17,4495	5,3602	1,5696	0,7039	0,8099
11	206,8056	201,2999	212,3113	5,2856	1,9538	7,8907	18,3495	0,4929	0,3929	0,8842
12	213,6364	209,2398	218,0330	4,8907	0,1058	5,0318	2543,7096	0,0288	0,0288	0,9905
13	222,0345	217,0550	227,0139	3,9696	1,8636	6,4544	13,4946	0,6260	0,4595	0,8672
14	210,1809	205,6050	214,7568	4,9353	0,3864	5,4505	223,8428	0,1044	0,1025	0,9670
15	211,6691	207,2515	216,0867	5,0364	0,0326	5,0800	27252,1555	0,0086	0,0086	0,9971
16	203,4681	198,6019	208,3343	5,7244	0,3298	6,1640	393,0909	0,0768	0,0760	0,9753
17	209,0353	203,7114	214,3592	5,0740	1,7281	7,3782	20,5073	0,4541	0,3708	0,8900

Tabela 7.10: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 80% de dados faltantes MCAR (3 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
31,323	52,317	30,995	53,709	35,541	49,284	37,187	51,658	33,171	57,031

A Tabela 7.11 traz os resultados obtidos com a biclusterização para a base de dados com 80% de dados faltantes, mas desta vez utilizando 10 imputações. Os graus de liberdade (*df*) são maiores para 15 dos 17 atributos quando se utilizam 10 em vez de 3 imputações. Ao contrário, os valores de *r* e γ não diminuíram para a maioria dos atributos. O valor de *r* foi melhor para 7 dos 17 atributos e o valor de γ foi melhor para 8 dos 17 atributos. Como esperado, com 10 imputações

o valor da *Eficiência* aumenta. Entretanto, a média real dos atributos da base de dados Yeast ainda não pertence ao intervalo estabelecido por uma confiança de 95%.

A Tabela 7.12 exibe os resultados da estimação dos dados faltantes utilizando MI com 10 imputações para a base de dados Yeast com 80% de dados faltantes MCAR. Os resultados foram ainda melhores que os obtidos com 3 imputações. Quando comparados com os resultados da heurística sendo executada apenas uma única vez, o ganho no valor de MAE foi de 9,66% e o ganho no valor do RMSE foi de 13,13%.

Tabela 7.11: Resultado da MI utilizando biclusterização – Yeast com 80% de dados faltantes MCAR (10 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	213,3514	208,0178	218,6850	3,5966	3,4622	7,4050	34,0251	1,0589	0,5405	0,9487
2	205,8177	200,5992	211,0363	4,2249	2,6038	7,0890	55,1343	0,6779	0,4245	0,9593
3	194,6145	188,9922	200,2368	4,8952	3,0302	8,2284	54,8458	0,6809	0,4257	0,9592
4	204,3839	199,6805	209,0873	4,3098	1,3171	5,7586	142,1876	0,3362	0,2619	0,9745
5	201,5615	197,1221	206,0009	4,5310	0,5448	5,1303	659,5861	0,1323	0,1195	0,9882
6	207,4513	202,9717	211,9310	4,0887	1,0318	5,2237	190,6532	0,2776	0,2254	0,9780
7	203,6999	198,9686	208,4312	4,4099	1,2883	5,8271	152,1637	0,3214	0,2530	0,9753
8	202,9473	198,0640	207,8306	4,3736	1,6671	6,2074	103,1273	0,4193	0,3087	0,9701
9	208,0660	203,2380	212,8940	4,0225	1,8593	6,0678	79,2174	0,5084	0,3532	0,9659
10	172,2859	166,1550	178,4168	5,7514	3,6664	9,7845	52,9718	0,7012	0,4332	0,9585
11	196,5767	191,7377	201,4158	4,8432	1,1384	6,0955	213,2404	0,2586	0,2128	0,9792
12	201,7621	197,0172	206,5071	4,6003	1,1458	5,8607	194,5945	0,2740	0,2230	0,9782
13	209,9907	205,2487	214,7327	4,0572	1,6330	5,8534	95,5722	0,4427	0,3209	0,9689
14	198,1337	193,4193	202,8482	4,6489	1,0334	5,7856	233,1483	0,2445	0,2033	0,9801
15	198,5513	194,1914	202,9112	4,6530	0,2684	4,9482	2528,7431	0,0634	0,0604	0,9940
16	193,5497	188,3464	198,7530	5,0320	1,8324	7,0477	110,0292	0,4006	0,2986	0,9710
17	195,8731	190,8913	200,8548	4,8080	1,5020	6,4602	137,5959	0,3436	0,2663	0,9741

Tabela 7.12: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 80% de dados faltantes MCAR (10 imputações).

Biclusterização					
Média		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
29,967	49,545	29,243	50,658	33,171	57,031

7.1.2 Dados faltantes segundo o mecanismo MAR

As bases de dados com 40% e 80% de dados faltantes, segundo o mecanismo MAR, utilizadas na Subseção 6.1.6 foram utilizadas neste experimento. Como descrito na Subseção

6.1.6, esses conjuntos de dados possuem dados faltantes nos atributos: 3, 5, 11 e 16. Os parâmetros utilizados para a execução do SwarmBcluster foram os mesmos adotados na Subseção 6.1.6.

Tabela 7.13: Resultados da MI utilizando biclusterização – Yeast com 40% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
3	208,9239	204,4715	213,3764	5,1365	0,0199	5,1604	1,8690E+05	0,0046	0,0046	0,9991
5	211,7801	207,4890	216,0712	4,7839	0,0078	4,7933	1,0430E+06	0,0020	0,0020	0,9996
11	209,3864	204,9876	213,7853	5,0334	0,0030	5,0370	7,8059E+06	0,0007	0,0007	0,9999
16	203,8336	199,2916	208,3756	5,3652	0,0040	5,3700	4,9809E+06	0,0009	0,0009	0,9998

Tabela 7.14: Resultados da MI utilizando NORM – Yeast com 40% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
3	208,2684	203,7898	212,7471	5,1311	0,0751	5,2213	1,3412E+04	0,0176	0,0174	0,9965
5	210,8160	206,4628	215,1691	4,8767	0,0468	4,9328	3,0916E+04	0,0115	0,0114	0,9977
11	208,8744	204,3562	213,3926	5,2040	0,0916	5,3139	9,3520E+03	0,0211	0,0209	0,9958
16	202,6897	197,9927	207,3866	5,5453	0,1645	5,7427	3,3863E+03	0,0356	0,0349	0,9931

As Tabelas 7.13 e 7.14 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 40% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 16. Logo, o número de iterações de cada ciclo do DA foi igual a 20, totalizando 100 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster foi superior ao NORM, uma vez que obteve resultados melhores para *r*, γ e *Efic*. Tanto para o NORM quanto para a biclusterização, a média real dos atributos pertence ao intervalo estabelecido por uma confiança de 95%.

A Tabela 7.15 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Yeast com 40% de dados faltantes MAR. Novamente, o SwarmBcluster produziu resultados melhores que os do NORM e mais uma vez tanto a imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que os obtidos com a imputação única (Subseção 6.1.6). O ganho no valor de MAE foi de 3,13% e o ganho no valor de RMSE foi de 4%.

Tabela 7.15: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 40% de dados faltantes MAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
16,883	22,995	16,959	23,137	22,243	29,264	23,117	30,370	17,428	23,952

Tabela 7.16: Resultados da MI utilizando biclusterização – Yeast com 80% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
3	208,3487	203,9964	212,7011	4,8779	0,0442	4,9310	3,4495E+04	0,0109	0,0108	0,9978
5	213,0819	208,8951	217,2687	4,5563	0,0055	4,5629	1,8777E+06	0,0015	0,0015	0,9997
11	210,7459	206,4544	215,0374	4,7780	0,0134	4,7940	3,5807E+05	0,0034	0,0033	0,9993
16	203,5361	199,0293	208,0429	5,2485	0,0323	5,2872	7,4543E+04	0,0074	0,0074	0,9985

Tabela 7.17: Resultados da MI utilizando NORM – Yeast com 80% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
3	208,6579	203,3407	213,9752	4,9921	1,9730	7,3597	3,8650E+01	0,4743	0,3543	0,9338
5	212,0151	207,5670	216,4632	4,7817	0,3071	5,1503	7,8104E+02	0,0771	0,0739	0,9854
11	208,6767	204,1645	213,1889	5,2247	0,0626	5,2998	1,9889E+04	0,0144	0,0143	0,9972
16	201,7994	196,2534	207,3453	5,3152	2,2428	8,0065	3,5401E+01	0,5063	0,3707	0,9310

As Tabelas 7.16 e 7.17 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 80% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 46. Logo, o número de iterações de cada ciclo do DA foi igual a 50, totalizando 250 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster foi superior ao NORM, principalmente para os atributos 3 e 16. Tanto para o NORM quanto para a biclusterização, a média real dos atributos pertence ao intervalo estabelecido por uma confiança de 95%.

A Tabela 7.18 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Yeast com 80% de dados faltantes MAR. O SwarmBcluster produziu novamente resultados melhores que os do NORM e melhores que os obtidos na Subseção 6.1.6. Porém, quando se compara os valores obtidos através da imputação pela média ou mediana das 5 imputações com os resultados obtidos na Subseção 6.1.6, percebe-se que o ganho foi pequeno. O ganho no valor de MAE foi de apenas 2,4% e o ganho no valor de RMSE foi de apenas 2,94%.

Tabela 7.18: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 80% de dados faltantes MAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
18,892	26,119	18,848	26,081	22,750	29,661	23,679	30,758	19,311	26,872

7.1.3 Dados faltantes segundo o mecanismo MNAR

As bases de dados com 40% e 80% de dados faltantes, segundo o mecanismo MNAR, utilizadas na Subseção 6.1.7 foram utilizadas neste experimento. Como descrito na Subseção 6.1.7, esses conjuntos de dados possuem dados faltantes no atributo 7. Os parâmetros adotados para a execução do SwarmBcluster foram os mesmos utilizados na Subseção 6.1.7.

As Tabelas 7.19 e 7.20 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 40% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 13. Logo, o número de iterações de cada ciclo do DA foi igual a 15, totalizando 75 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster foi superior ao NORM, como verifica-se através de r, γ e *Efic.*, sendo que as 5 imputações realizadas foram suficientes para o SwarmBcluster praticamente atingir a eficiência máxima. Tanto para o NORM quanto para a biclusterização, a média real do atributo 7 pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.19: Resultados da MI utilizando biclusterização – Yeast com 40% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
7	214,1426	209,9505	218,3347	4,5638	0,0090	4,5745	7,2116E+05	0,0024	0,0024	0,9995

Tabela 7.20: Resultados da MI utilizando NORM – Yeast com 40% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
7	213,2616	208,9981	217,5250	4,6868	0,0373	4,7316	4,4790E+04	0,0095	0,0095	0,9981

A Tabela 7.21 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Yeast com 40% de dados faltantes MNAR. Novamente, o SwarmBcluster produziu resultados melhores que os do NORM e, mais uma vez, tanto a imputação pela média como pela

mediana das 5 imputações de cada dado faltante produziram resultados melhores que os obtidos com a imputação única na Subseção 6.1.7. O ganho no valor de MAE foi de 1,95% e o ganho no valor de RMSE foi de 2,17%.

Tabela 7.21: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 40% de dados faltantes MNAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
16,881	22,928	17,017	23,098	18,259	23,357	18,804	23,860	17,217	23,436

As Tabelas 7.22 e 7.23 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 80% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 40. Logo, o número de iterações de cada ciclo do DA foi igual a 45, totalizando 225 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster foi superior ao NORM, como verifica-se através de r, γ e *Efic*. Tanto para o NORM quanto para a biclusterização, a média real do atributo 7 pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.22: Resultados da MI utilizando biclusterização – Yeast com 80% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
7	212,8879	208,6542	217,1216	4,6441	0,0182	4,6659	183530	0,0047	0,0047	0,9991

Tabela 7.23: Resultados da MI utilizando NORM – Yeast com 80% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
7	210,0719	205,5292	214,6146	5,0138	0,2984	5,3718	900,51	0,0714	0,0687	0,9864

Tabela 7.24: Resultados da estimação dos dados faltantes utilizando MI – Yeast com 80% de dados faltantes MNAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
18,181	24,538	18,216	24,647	20,137	25,776	20,893	26,624	18,535	25,167

A Tabela 7.24 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Yeast com 80% de dados faltantes MNAR. O SwarmBcluster produziu resultados melhores que os do NORM também nesse aspecto e, mais uma vez, tanto a imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que os obtidos com a imputação única na Subseção 6.1.7. O ganho no valor de MAE foi de 1,91% e o ganho no valor de RMSE foi de 2,5%.

7.2 Base de dados Human

Os experimentos listados a seguir foram realizados com a base de dados Human. Como o parâmetro de interesse escolhido para este trabalho é a média aritmética, a Tabela 7.25 exibe a média dos 40 atributos da base de dados Human.

Tabela 7.25: Média aritmética dos 40 atributos da base de dados Human.

Atributo	1	2	3	4	5	6	7	8	9	10
Média	0,2564	2,5803	9,3964	4,2703	13,3048	6,7367	7,1521	13,3036	1,1458	3,4936
Atributo	11	12	13	14	15	16	17	18	19	20
Média	1,9933	10,8991	10,467	15,5142	3,5024	10,5842	-4,0821	4,7412	-2,0142	2,9155
Atributo	21	22	23	24	25	26	27	28	29	30
Média	-3,4306	18,2855	10,5733	9,8818	6,6627	3,5918	3,1397	7,2148	6,6288	20,0339
Atributo	31	32	33	34	35	36	37	38	39	40
Média	-14,3848	-20,0252	6,9488	1,0073	4,2158	-6,09	-0,5639	-7,7261	-1,0376	1,643

A primeira parte desta seção traz os experimentos realizados com as bases de dados com 15%, 50% e 80% de dados faltantes segundo o mecanismo MCAR. A segunda parte traz os experimentos realizados com as bases de dados com 40% e 80% de dados faltantes segundo o mecanismo MAR. A terceira parte traz os experimentos realizados com as bases de dados com 40% e 80% de dados faltantes segundo o mecanismo MNAR.

7.2.1 Dados faltantes segundo o mecanismo MCAR

As bases de dados com 15%, 50% e 80% de dados faltantes, segundo o mecanismo MCAR, utilizadas nas Subseções 6.2.1 a 6.2.5 foram utilizadas neste experimento.

Tabela 7.26: Resultados da MI utilizando biclusterização – Human com 15% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	0,3982	-2,5096	3,3059	2,1922	0,0072	2,2009	257709,6720	0,0040	0,0039	0,9992
2	2,6793	-0,3337	5,6922	2,3467	0,0136	2,3630	83917,1768	0,0070	0,0069	0,9986
3	8,7908	6,8920	10,6895	0,9345	0,0033	0,9385	222181,9856	0,0043	0,0043	0,9992
4	4,4374	2,4084	6,4664	1,0536	0,0150	1,0716	14133,0575	0,0171	0,0170	0,9966
5	13,0030	11,1950	14,8111	0,8494	0,0013	0,8510	1142930,2834	0,0019	0,0019	0,9996
6	6,0010	4,1438	7,8582	0,8926	0,0044	0,8979	116405,9859	0,0059	0,0059	0,9988
7	6,9634	4,5044	9,4225	1,5701	0,0033	1,5740	628879,0704	0,0025	0,0025	0,9995
8	13,1631	10,5247	15,8015	1,8037	0,0070	1,8120	188234,1264	0,0046	0,0046	0,9991
9	1,1089	-1,8013	4,0191	2,1906	0,0116	2,2046	99499,5250	0,0064	0,0064	0,9987
10	3,4661	1,2990	5,6332	1,2167	0,0048	1,2224	181391,1703	0,0047	0,0047	0,9991
11	1,9617	-0,3930	4,3164	1,4338	0,0079	1,4433	92257,4321	0,0066	0,0066	0,9987
12	10,5582	8,4138	12,7026	1,1883	0,0072	1,1970	76410,8587	0,0073	0,0073	0,9985
13	9,7978	8,1748	11,4208	0,6720	0,0114	0,6857	10126,7908	0,0203	0,0201	0,9960
14	14,7576	12,9435	16,5717	0,8531	0,0030	0,8567	225926,3602	0,0042	0,0042	0,9992
15	3,6885	1,5930	5,7841	1,1408	0,0019	1,1431	963058,9574	0,0020	0,0020	0,9996
16	11,4419	8,8162	14,0676	1,7661	0,0237	1,7946	15874,6073	0,0161	0,0160	0,9968
17	-3,7078	-6,3960	-1,0196	1,8682	0,0108	1,8811	84995,4512	0,0069	0,0069	0,9986
18	5,0782	2,4477	7,7087	1,7944	0,0057	1,8012	274182,9582	0,0038	0,0038	0,9992
19	-2,7726	-5,6227	0,0775	2,1092	0,0045	2,1145	624665,6873	0,0025	0,0025	0,9995
20	3,0001	0,8418	5,1584	1,1666	0,0383	1,2126	2781,8460	0,0394	0,0386	0,9923
21	-3,2856	-5,3137	-1,2574	1,0651	0,0047	1,0707	146169,8783	0,0053	0,0052	0,9990
22	18,1196	15,4663	20,7728	1,8240	0,0070	1,8325	187745,7323	0,0046	0,0046	0,9991
23	9,8598	7,7358	11,9837	1,1683	0,0050	1,1743	151514,6334	0,0052	0,0052	0,9990
24	9,3818	7,1411	11,6225	1,3057	0,0010	1,3069	4481637,3655	0,0009	0,0009	0,9998
25	6,9045	4,3416	9,4674	1,7043	0,0046	1,7098	385307,5163	0,0032	0,0032	0,9994
26	3,4536	1,7560	5,1512	0,7479	0,0019	0,7502	417720,9804	0,0031	0,0031	0,9994
27	3,5771	1,4320	5,7221	1,1889	0,0073	1,1977	74239,7718	0,0074	0,0074	0,9985
28	6,8141	4,5236	9,1047	1,3474	0,0152	1,3657	22291,2297	0,0136	0,0135	0,9973
29	6,5478	4,5618	8,5339	1,0163	0,0087	1,0268	38646,9104	0,0103	0,0102	0,9980
30	20,0576	16,2424	23,8728	3,7668	0,0185	3,7890	115975,9324	0,0059	0,0059	0,9988
31	-13,5576	-17,2643	-9,8509	3,5422	0,0286	3,5765	43288,9337	0,0097	0,0097	0,9981
32	-19,0971	-22,0442	-16,1501	2,2556	0,0043	2,2608	755597,7853	0,0023	0,0023	0,9995
33	6,4158	4,4750	8,3566	0,9784	0,0018	0,9805	850494,5151	0,0022	0,0022	0,9996
34	0,5402	-1,5284	2,6088	1,1095	0,0037	1,1139	252998,6749	0,0040	0,0040	0,9992
35	3,8602	1,8807	5,8397	1,0158	0,0035	1,0200	233235,5787	0,0042	0,0041	0,9992
36	-6,2734	-8,6337	-3,9131	1,4464	0,0032	1,4502	585249,1381	0,0026	0,0026	0,9995
37	-1,1813	-3,8753	1,5127	1,8607	0,0238	1,8893	17529,9039	0,0153	0,0152	0,9970
38	-7,5566	-10,3360	-4,7772	2,0063	0,0039	2,0109	756104,8951	0,0023	0,0023	0,9995
39	-1,2147	-3,6544	1,2250	1,5301	0,0160	1,5494	25913,7401	0,0126	0,0125	0,9975
40	1,7812	0,2916	3,2707	0,5772	0,0003	0,5776	7670633,5067	0,0007	0,0007	0,9999

Tabela 7.27: Resultados da MI utilizando NORM – Human com 15% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	1,0250	-2,1483	4,1982	2,3044	0,2640	2,6212	273,8552	0,1375	0,1272	0,9752
2	1,9113	-1,5030	5,3255	2,4982	0,4469	3,0345	128,0848	0,2147	0,1893	0,9635
3	8,8289	6,5008	11,1571	1,0164	0,3288	1,4110	51,1504	0,3882	0,3062	0,9423
4	4,3567	2,1587	6,5547	1,1291	0,1071	1,2576	383,1827	0,1138	0,1068	0,9791
5	13,0819	10,8855	15,2784	0,9157	0,2834	1,2558	54,5389	0,3714	0,2962	0,9441
6	6,3049	4,3725	8,2373	0,9388	0,0277	0,9721	3413,0519	0,0354	0,0348	0,9931
7	7,1764	4,5775	9,7752	1,6676	0,0754	1,7581	1509,0924	0,0543	0,0527	0,9896
8	12,7982	9,9449	15,6516	1,8935	0,1882	2,1193	352,3155	0,1193	0,1116	0,9782
9	1,6804	-1,5563	4,9170	2,2735	0,3779	2,7270	144,6673	0,1994	0,1776	0,9657
10	3,4487	1,0458	5,8515	1,3293	0,1447	1,5030	299,6568	0,1306	0,1214	0,9763
11	1,4150	-1,3759	4,2060	1,5542	0,3946	2,0277	73,3500	0,3047	0,2536	0,9517
12	11,5254	9,1447	13,9061	1,2636	0,1765	1,4754	194,1521	0,1676	0,1522	0,9705
13	10,4533	8,6737	12,2329	0,7118	0,0938	0,8244	214,3664	0,1582	0,1445	0,9719
14	15,1967	13,2692	17,1242	0,9149	0,0435	0,9671	1374,6954	0,0570	0,0553	0,9891
15	3,2830	0,7742	5,7919	1,2334	0,3376	1,6385	65,4349	0,3285	0,2692	0,9489
16	10,5645	7,6684	13,4606	1,8961	0,2393	2,1833	231,1456	0,1515	0,1390	0,9730
17	-3,9334	-6,9446	-0,9221	1,9877	0,3106	2,3604	160,4125	0,1875	0,1682	0,9675
18	4,8352	2,1134	7,5569	1,8953	0,0275	1,9283	13674,9833	0,0174	0,0172	0,9966
19	-2,4044	-5,3407	0,5320	2,2152	0,0243	2,2444	23632,7351	0,0132	0,0131	0,9974
20	2,6432	0,4062	4,8803	1,2172	0,0712	1,3027	929,8328	0,0702	0,0676	0,9867
21	-3,8270	-6,0247	-1,6294	1,1341	0,1025	1,2572	417,5611	0,1085	0,1022	0,9800
22	18,7682	15,9058	21,6306	1,9027	0,1918	2,1328	343,4831	0,1210	0,1131	0,9779
23	10,1394	7,8525	12,4263	1,2508	0,0922	1,3614	606,2345	0,0884	0,0842	0,9834
24	10,3243	7,9080	12,7406	1,3848	0,1124	1,5198	507,3823	0,0974	0,0924	0,9819
25	6,8528	4,0085	9,6972	1,8545	0,2095	2,1060	280,5857	0,1356	0,1256	0,9755
26	3,4034	1,5656	5,2412	0,8065	0,0606	0,8792	585,2423	0,0901	0,0858	0,9831
27	3,5398	1,1544	5,9252	1,2539	0,1894	1,4812	169,8577	0,1813	0,1633	0,9684
28	7,5047	5,0998	9,9096	1,4372	0,0569	1,5055	1943,4505	0,0475	0,0463	0,9908
29	7,1613	5,0969	9,2256	1,0934	0,0133	1,1093	19404,1293	0,0146	0,0145	0,9971
30	20,4786	16,5303	24,4269	3,9604	0,0813	4,0580	6913,0135	0,0246	0,0243	0,9952
31	-13,7468	-17,9236	-9,5701	3,6533	0,7399	4,5412	104,6434	0,2430	0,2105	0,9596
32	-19,2709	-22,4054	-16,1364	2,3700	0,1563	2,5575	744,1246	0,0791	0,0758	0,9851
33	6,7639	4,7275	8,8002	1,0509	0,0238	1,0794	5730,0944	0,0271	0,0268	0,9947
34	0,8796	-1,3355	3,0947	1,1763	0,0841	1,2772	640,4473	0,0858	0,0819	0,9839
35	3,8482	1,7457	5,9506	1,0870	0,0530	1,1506	1308,8565	0,0585	0,0567	0,9888
36	-6,8847	-9,4030	-4,3665	1,5561	0,0789	1,6508	1215,4588	0,0609	0,0589	0,9884
37	-0,7892	-3,7120	2,1337	1,9291	0,2457	2,2238	227,6385	0,1528	0,1401	0,9727
38	-7,4818	-10,4307	-4,5330	2,0892	0,1453	2,2636	674,0849	0,0835	0,0798	0,9843
39	-0,5629	-3,1042	1,9784	1,6091	0,0600	1,6811	2180,3267	0,0447	0,0437	0,9913
40	2,0713	0,3068	3,8358	0,6275	0,1525	0,8105	78,4648	0,2916	0,2448	0,9533

Devido ao melhor tempo de execução e à qualidade dos resultados obtidos pela heurística na Subseção 6.2.4, o SwarmBcluster foi parametrizado como naquele experimento, com $\delta = 600$

para as bases com 15% e 50% de dados faltantes, e com $\delta = 1200$ para a base com 80% de dados faltantes.

As Tabelas 7.26 e 7.27 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 15% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 20. Logo, o número de iterações de cada ciclo do DA foi igual a 25, totalizando 125 iterações para produzir os cinco conjuntos de dados completos. Apesar do NORM ter obtido bons resultados para *df*, os resultados produzidos pela técnica de biclusterização foram superiores, indicando que a variância total foi bem estimada. A biclusterização também obteve resultados melhores para *r*, o que indica maior estabilidade nos parâmetros estimados, consequentemente refletindo em uma taxa menor de informação faltante (γ). A biclusterização também foi superior quanto à eficiência das estimativas obtidas com 5 imputações quando comparadas a um número infinito de imputações (*Efic.*), o que indica que pouco se ganharia aumentando o número de imputações. Tanto para o NORM quanto para a biclusterização, a média real dos atributos da base de dados Human pertence ao intervalo estabelecido por uma confiança de 95%.

A Tabela 7.28 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Human com 15% de dados faltantes MCAR. Além de o SwarmBcluster ter produzido resultados consideravelmente melhores que os do NORM, tanto a imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que a imputação baseada em uma única execução da heurística. O ganho no valor de MAE é de 3,59% e o ganho no valor de RMSE é de 3,92%.

Tabela 7.28: Resultados da estimação dos dados faltantes utilizando MI – Human com 15% de dados faltantes MCAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
29,928	41,537	29,971	41,633	41,128	56,077	42,892	58,263	31,043	43,233

Tabela 7.29: Resultados da MI utilizando biclusterização – Human com 50% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	1,3077	-1,5327	4,1481	1,9293	0,1424	2,1002	604,2458	0,0886	0,0844	0,9834
2	3,3389	0,5351	6,1428	1,9495	0,0808	2,0464	1780,9770	0,0497	0,0485	0,9904
3	8,4758	6,7360	10,2157	0,7434	0,0372	0,7880	1249,5101	0,0600	0,0581	0,9885
4	4,4247	2,6281	6,2213	0,8208	0,0162	0,8402	7495,3322	0,0236	0,0234	0,9953
5	11,0987	9,4574	12,7399	0,6782	0,0192	0,7012	3711,3350	0,0339	0,0334	0,9934
6	5,3968	3,6357	7,1580	0,7747	0,0272	0,8074	2442,2829	0,0422	0,0413	0,9918
7	6,3222	4,0209	8,6234	1,3435	0,0292	1,3785	6178,0757	0,0261	0,0258	0,9949
8	12,3652	9,7778	14,9526	1,5695	0,1444	1,7427	404,7548	0,1104	0,1038	0,9797
9	1,2943	-1,4470	4,0357	1,9100	0,0385	1,9563	7154,1524	0,0242	0,0239	0,9952
10	3,8874	1,9162	5,8585	0,9838	0,0230	1,0114	5375,7888	0,0280	0,0276	0,9945
11	1,7611	-0,4636	3,9858	1,2165	0,0599	1,2884	1285,8548	0,0591	0,0572	0,9887
12	8,0923	6,0659	10,1186	0,9671	0,0848	1,0689	441,0637	0,1053	0,0993	0,9805
13	9,1312	7,6837	10,5787	0,5212	0,0201	0,5454	2036,6645	0,0464	0,0453	0,9910
14	12,0455	10,2599	13,8312	0,7231	0,0891	0,8300	241,0830	0,1479	0,1359	0,9735
15	5,0447	3,1943	6,8950	0,8765	0,0123	0,8913	14569,2081	0,0168	0,0167	0,9967
16	9,4293	6,9825	11,8761	1,5338	0,0205	1,5584	16068,6268	0,0160	0,0159	0,9968
17	-3,1861	-5,7494	-0,6229	1,5758	0,1121	1,7103	646,9356	0,0853	0,0815	0,9840
18	4,5951	2,1255	7,0647	1,4741	0,0946	1,5875	782,7572	0,0770	0,0738	0,9854
19	-1,2284	-3,9380	1,4812	1,8218	0,0745	1,9112	1828,7222	0,0491	0,0478	0,9905
20	2,9925	0,9602	5,0249	0,9793	0,0799	1,0752	502,5419	0,0980	0,0928	0,9818
21	-3,2355	-5,1317	-1,3393	0,8807	0,0460	0,9359	1147,4551	0,0627	0,0607	0,9880
22	17,7258	15,1317	20,3199	1,6439	0,0898	1,7517	1056,8366	0,0656	0,0633	0,9875
23	8,6925	6,7233	10,6618	0,9851	0,0203	1,0095	6876,7323	0,0247	0,0244	0,9951
24	8,5684	6,4864	10,6505	1,0959	0,0271	1,1284	4814,9767	0,0297	0,0292	0,9942
25	6,2217	3,8634	8,5801	1,4311	0,0139	1,4478	30161,1321	0,0117	0,0116	0,9977
26	4,1917	2,6399	5,7435	0,5755	0,0427	0,6268	597,2837	0,0891	0,0849	0,9833
27	5,0430	3,0530	7,0330	1,0197	0,0093	1,0309	34365,1176	0,0109	0,0108	0,9978
28	5,6710	3,5260	7,8160	1,1372	0,0504	1,1977	1568,4757	0,0532	0,0517	0,9898
29	4,5572	2,7695	6,3449	0,8190	0,0108	0,8319	16547,8162	0,0158	0,0157	0,9969
30	18,8881	15,1873	22,5889	3,5183	0,0390	3,5652	23162,1230	0,0133	0,0132	0,9974
31	-12,7730	-16,3870	-9,1590	3,2304	0,1412	3,3999	1609,5769	0,0525	0,0510	0,9899
32	-15,6851	-18,5177	-12,8524	1,8767	0,1767	2,0887	388,2697	0,1130	0,1061	0,9792
33	6,0881	4,2428	7,9333	0,8256	0,0506	0,8863	853,0825	0,0735	0,0707	0,9861
34	0,6370	-1,3833	2,6573	0,9833	0,0660	1,0625	719,9996	0,0805	0,0771	0,9848
35	1,9653	0,1394	3,7913	0,8299	0,0317	0,8679	2083,3451	0,0458	0,0447	0,9911
36	-7,0672	-9,2873	-4,8470	1,2472	0,0299	1,2830	5131,0363	0,0287	0,0283	0,9944
37	-0,4869	-3,0707	2,0969	1,6686	0,0577	1,7378	2520,5840	0,0415	0,0406	0,9919
38	-7,1388	-9,7313	-4,5464	1,7195	0,0250	1,7495	13615,2404	0,0174	0,0173	0,9966
39	-0,8679	-3,3473	1,6115	1,3053	0,2457	1,6002	117,7814	0,2259	0,1978	0,9619
40	1,7507	0,3826	3,1189	0,4699	0,0145	0,4873	3157,0256	0,0369	0,0362	0,9928

Tabela 7.30: Resultados da MI utilizando NORM – Human com 50% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
1	0,9733	-2,7699	4,7164	2,3390	1,0902	3,6472	31,0903	0,5593	0,3963	0,9266
2	2,7830	-1,3418	6,9078	2,3714	1,7146	4,4289	18,5342	0,8676	0,5143	0,9067
3	9,1724	6,7597	11,5851	1,0167	0,4155	1,5153	36,9423	0,4904	0,3627	0,9324
4	4,2540	2,0327	6,4753	1,0687	0,1798	1,2844	141,7894	0,2019	0,1795	0,9654
5	12,7986	10,4385	15,1587	0,9342	0,4298	1,4500	31,6147	0,5521	0,3929	0,9271
6	6,9452	3,6039	10,2864	0,9217	1,6536	2,9061	8,5791	2,1528	0,7376	0,8714
7	7,0557	3,3973	10,7140	1,7091	1,4790	3,4838	15,4131	1,0384	0,5627	0,8988
8	14,6026	10,7205	18,4847	1,9742	1,6240	3,9230	16,2095	0,9871	0,5492	0,9010
9	1,1834	-2,4334	4,8002	2,2385	0,9722	3,4052	34,0764	0,5212	0,3781	0,9297
10	5,0542	2,1420	7,9664	1,3247	0,7358	2,2076	25,0085	0,6665	0,4428	0,9186
11	1,5428	-1,6788	4,7644	1,5470	0,9622	2,7017	21,8984	0,7464	0,4734	0,9135
12	9,6404	6,6754	12,6053	1,2302	0,8818	2,2883	18,7069	0,8602	0,5119	0,9071
13	10,8705	8,8534	12,8876	0,6981	0,3008	1,0591	34,4370	0,5170	0,3760	0,9301
14	14,1459	12,1342	16,1576	0,9042	0,1244	1,0534	199,2603	0,1651	0,1502	0,9708
15	5,4156	2,8745	7,9568	1,1424	0,4488	1,6809	38,9735	0,4714	0,3527	0,9341
16	9,9657	6,4440	13,4874	1,9083	1,1001	3,2285	23,9221	0,6918	0,4528	0,9170
17	-3,2001	-6,3558	-0,0445	2,0231	0,4742	2,5922	82,9973	0,2813	0,2377	0,9546
18	3,8638	0,8957	6,8318	1,7654	0,4398	2,2932	75,5158	0,2990	0,2498	0,9524
19	-0,1483	-3,7258	3,4292	2,2732	0,8820	3,3315	39,6354	0,4656	0,3497	0,9346
20	2,8088	0,5130	5,1046	1,1989	0,1443	1,3720	251,2297	0,1444	0,1331	0,9741
21	-3,9253	-6,4083	-1,4422	1,1304	0,3955	1,6050	45,7513	0,4198	0,3246	0,9390
22	18,1754	14,4190	21,9318	1,9822	1,4091	3,6732	18,8754	0,8530	0,5097	0,9075
23	12,0167	9,2307	14,8027	1,3047	0,5965	2,0204	31,8707	0,5486	0,3913	0,9274
24	8,9864	6,2589	11,7139	1,3700	0,4721	1,9365	46,7414	0,4135	0,3210	0,9397
25	5,3989	1,4866	9,3112	1,8471	1,7811	3,9844	13,9013	1,1571	0,5913	0,8943
26	3,6435	1,5714	5,7156	0,7873	0,2752	1,1176	45,8026	0,4195	0,3244	0,9391
27	4,2099	0,9172	7,5027	1,3041	1,2652	2,8223	13,8224	1,1643	0,5929	0,8940
28	7,7681	5,2712	10,2649	1,4210	0,1682	1,6228	258,6192	0,1420	0,1311	0,9745
29	5,5708	3,1482	7,9935	1,1116	0,3469	1,5278	53,8810	0,3745	0,2980	0,9437
30	19,3081	14,5988	24,0175	4,2994	1,2281	5,7731	61,3818	0,3428	0,2784	0,9473
31	-12,1900	-17,9612	-6,4188	3,7983	4,0599	8,6702	12,6683	1,2827	0,6178	0,8900
32	-19,2975	-23,3130	-15,2819	2,2675	1,6083	4,1974	18,9209	0,8511	0,5091	0,9076
33	5,1048	2,8535	7,3562	1,0819	0,1979	1,3194	123,4267	0,2195	0,1930	0,9628
34	1,8390	-1,0464	4,7244	1,1994	0,8065	2,1672	20,0567	0,8070	0,4946	0,9100
35	3,1763	1,0583	5,2943	1,0748	0,0774	1,1677	632,1069	0,0864	0,0824	0,9838
36	-7,9776	-11,1547	-4,8006	1,5975	0,8584	2,6275	26,0272	0,6448	0,4339	0,9201
37	1,6235	-1,9401	5,1870	2,0197	1,0717	3,3057	26,4297	0,6367	0,4306	0,9207
38	-6,9194	-10,2405	-3,5983	2,0643	0,6723	2,8711	50,6556	0,3908	0,3078	0,9420
39	-1,5078	-4,6658	1,6501	1,6208	0,8127	2,5960	28,3436	0,6017	0,4155	0,9233
40	2,5667	0,5069	4,6266	0,6322	0,3936	1,1045	21,8781	0,7470	0,4736	0,9135

As Tabelas 7.29 e 7.30 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 50% de dados faltantes. O número de imputações foi

igual a 5. O algoritmo EM convergiu na iteração 283. Logo, o número de iterações de cada ciclo do DA foi igual a 285, totalizando 1.425 iterações para produzir os cinco conjuntos de dados completos. Os resultados desse teste foram parecidos com o caso de 15% de dados faltantes. A biclusterização também obteve melhores resultados para *r*, refletindo em uma taxa menor de informação faltante (γ), o que significa maior estabilidade nos valores estimados. No caso da biclusterização, a *Eficiência* foi superior a 96% para todos os atributos. Já para o NORM, alguns atributos ficaram com *Eficiência* inferior a 90%. Apesar da maior estabilidade nos valores estimados pelo SwarmBcluster, os intervalos de confiança produzidos pelo NORM foram melhores. Para o NORM, a média real de todos os atributos da base de dados Human pertence ao intervalo estabelecido por uma confiança de 95%. Para o SwarmBcluster, a média real de 34 dos 40 atributos da base de dados Human pertence ao intervalo estabelecido por uma confiança de 95%. Mais uma vez, o êxito do NORM pode estar relacionado ao fato de que ele foi construído para estimar parâmetros de interesse sob uma distribuição normal multivariada.

A Tabela 7.31 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Human com 50% de dados faltantes MCAR. O SwarmBcluster foi consideravelmente melhor que o NORM nas imputações realizadas com base nas MI. Mais uma vez, tanto a imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que a imputação baseada em uma única execução da heurística. Os ganhos em termos de qualidade com o emprego de MI foram maiores que os obtidos no teste com a base de dados com 15% de dados faltantes. O ganho no valor de MAE foi de 4,21% e o ganho no valor de RMSE foi de 5,31%.

Tabela 7.31: Resultados da estimação dos dados faltantes utilizando MI – Human com 50% de dados faltantes MCAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
32,362	44,604	32,322	44,550	45,221	62,227	47,167	64,414	33,741	47,046

Tabela 7.32: Resultados da MI utilizando biclusterização – Human com 80% de dados faltantes MCAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*R*	γ	*Efic.*
1	1,2162	-2,3075	4,7399	1,3418	1,5752	3,2321	11,6944	1,4087	0,6414	0,8863
2	0,3445	-2,3494	3,0384	1,4152	0,3949	1,8891	63,5705	0,3348	0,2734	0,9482
3	5,3466	3,9135	6,7797	0,3854	0,1244	0,5346	51,3390	0,3872	0,3057	0,9424
4	4,2359	2,8930	5,5787	0,4447	0,0205	0,4694	1449,8860	0,0554	0,0538	0,9893
5	7,9562	6,1230	9,7894	0,4144	0,3836	0,8748	14,4444	1,1108	0,5806	0,8960
6	4,8239	3,2952	6,3527	0,5552	0,0443	0,6084	524,6755	0,0957	0,0908	0,9822
7	5,8371	3,9205	7,7538	0,7831	0,1443	0,9563	122,0228	0,2211	0,1942	0,9626
8	8,1243	5,7913	10,4572	0,9524	0,3870	1,4168	37,2231	0,4877	0,3612	0,9326
9	1,2649	-1,3256	3,8555	1,1644	0,4854	1,7470	35,9741	0,5003	0,3677	0,9315
10	4,1321	2,5098	5,7544	0,5776	0,0896	0,6851	162,5032	0,1861	0,1671	0,9677
11	1,6886	-0,5137	3,8908	0,6247	0,5315	1,2625	15,6728	1,0210	0,5582	0,8996
12	8,2261	6,6592	9,7930	0,5558	0,0694	0,6391	235,6234	0,1498	0,1376	0,9732
13	5,7897	4,0810	7,4983	0,3233	0,3639	0,7600	12,1179	1,3504	0,6308	0,8880
14	11,1475	9,3944	12,9006	0,4894	0,2589	0,8000	26,5272	0,6348	0,4297	0,9209
15	4,0623	1,7852	6,3393	0,6431	0,5889	1,3497	14,5934	1,0988	0,5777	0,8964
16	10,3623	7,6472	13,0774	0,9046	0,8452	1,9189	14,3174	1,1212	0,5830	0,8956
17	-0,6088	-2,8422	1,6246	0,9033	0,3293	1,2984	43,1947	0,4374	0,3344	0,9373
18	6,3208	4,0232	8,6184	0,9395	0,3622	1,3742	39,9856	0,4626	0,3481	0,9349
19	-0,6198	-3,0296	1,7900	1,1892	0,2687	1,5117	87,9313	0,2711	0,2306	0,9559
20	3,8839	2,2795	5,4883	0,6038	0,0552	0,6701	409,3028	0,1097	0,1032	0,9798
21	-0,8411	-2,9379	1,2558	0,6387	0,4215	1,1445	20,4807	0,7919	0,4895	0,9108
22	13,7885	11,5478	16,0292	1,0820	0,1874	1,3069	135,0855	0,2078	0,1841	0,9645
23	7,2203	4,8504	9,5903	0,6563	0,6714	1,4621	13,1703	1,2277	0,6066	0,8918
24	7,9603	6,3491	9,5715	0,6571	0,0155	0,6757	5264,7852	0,0283	0,0279	0,9944
25	5,1429	3,2425	7,0432	0,7973	0,1189	0,9401	173,4957	0,1790	0,1615	0,9687
26	3,8995	2,4165	5,3825	0,3327	0,1998	0,5725	22,7994	0,7208	0,4639	0,9151
27	4,7166	2,9044	6,5288	0,5748	0,2334	0,8549	37,2538	0,4874	0,3611	0,9326
28	3,4188	1,5494	5,2882	0,6910	0,1823	0,9097	69,1906	0,3166	0,2615	0,9503
29	3,2455	1,6403	4,8506	0,5018	0,1408	0,6707	63,0403	0,3367	0,2746	0,9479
30	17,6698	13,8717	21,4678	2,3466	1,1736	3,7550	28,4354	0,6002	0,4148	0,9234
31	-12,7744	-16,9250	-8,6239	2,5479	1,6138	4,4844	21,4495	0,7601	0,4783	0,9127
32	-11,1058	-14,6177	-7,5939	1,3590	1,5428	3,2104	12,0277	1,3623	0,6330	0,8876
33	4,0091	2,2546	5,7636	0,4881	0,2610	0,8013	26,1828	0,6417	0,4326	0,9204
34	1,1184	-0,7468	2,9836	0,6525	0,2109	0,9056	51,2213	0,3878	0,3060	0,9423
35	1,9949	0,2759	3,7138	0,5018	0,2228	0,7692	33,1117	0,5327	0,3837	0,9287
36	-3,2845	-5,1237	-1,4453	0,7296	0,1258	0,8805	136,1739	0,2068	0,1833	0,9646
37	-3,4571	-5,6276	-1,2865	1,0196	0,1723	1,2264	140,6958	0,2028	0,1802	0,9652
38	-5,5700	-8,4999	-2,6400	1,1396	0,9126	2,2347	16,6571	0,9609	0,5419	0,9022
39	-0,4749	-2,7449	1,7952	0,9252	0,3468	1,3414	41,5568	0,4498	0,3412	0,9361
40	2,6211	1,1931	4,0492	0,2806	0,2085	0,5308	18,0072	0,8915	0,5216	0,9055

A Tabela 7.32 traz os resultados obtidos pela biclusterização para a base de dados com 80% de dados faltantes. Para esse conjunto de dados, não existem resultados para o NORM porque já

no primeiro ciclo ele retornou um erro devido à matriz de covariância ter se tornado não-definida positiva. Como já era esperado, a biclusterização produziu resultados consideravelmente piores que nos dois primeiros testes, devido à grande quantidade de dados faltantes. A informação faltante estimada (γ) foi maior que 50% para vários atributos. Apenas para 21 atributos, a média real dos atributos da base de dados Human pertence ao intervalo estabelecido por uma confiança de 95%.

A Tabela 7.33 exibe os resultados da estimação dos dados faltantes utilizando MI com 5 imputações para a base de dados Human com 80% de dados faltantes MCAR. Mais uma vez tanto a imputação utilizando a média como a mediana das 5 imputações de cada dado faltante produziram resultados melhores que a heurística executada uma única vez. O ganho no valor de MAE foi de 4,32% e o ganho no valor de RMSE foi de 6,01%.

Tabela 7.33: Resultados da estimação dos dados faltantes utilizando MI – Human com 80% de dados faltantes MCAR (5 imputações).

Média		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
39,723	55,050	39,905	55,315	41,518	58,571

7.2.2 Dados faltantes segundo o mecanismo MAR

As bases de dados com 40% e 80% de dados faltantes, segundo o mecanismo MAR, utilizadas na Subseção 6.2.6 foram utilizadas neste experimento. Como descrito na Subseção 6.2.6, esses conjuntos de dados possuem dados faltantes nos atributos: 22, 25, 34 e 40. Os parâmetros utilizados para a execução do SwarmBcluster foram os mesmos adotados na Subseção 6.2.6.

As Tabelas 7.34 e 7.35 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 40% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 10. Logo, o número de iterações de cada ciclo do DA foi igual a 15, totalizando 75 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster foi superior ao NORM, uma vez que obteve resultados melhores para *r*, γ e *Efic*. Tanto para o NORM quanto para a biclusterização, a média real dos atributos pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.34: Resultados da MI utilizando biclusterização – Human com 40% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
22	19,1826	16,5193	21,8459	1,8218	0,0205	1,8464	22430,4992	0,0135	0,0134	0,9973
25	5,9151	3,3268	8,5035	1,7364	0,0063	1,7439	213091,8435	0,0044	0,0043	0,9991
34	1,2727	-0,8354	3,3809	1,1271	0,0249	1,1569	6020,0170	0,0265	0,0261	0,9948
40	1,6710	0,1511	3,1908	0,5982	0,0026	0,6013	150205,9014	0,0052	0,0052	0,9990

Tabela 7.35: Resultados da MI utilizando NORM – Human com 40% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
22	19,0679	16,2358	21,9001	1,9416	0,1220	2,0879	814,2142	0,0754	0,0724	0,9857
25	6,7680	4,0480	9,4880	1,8446	0,0677	1,9259	2244,5994	0,0441	0,0431	0,9915
34	0,5484	-1,6683	2,7652	1,1904	0,0740	1,2791	830,4753	0,0746	0,0716	0,9859
40	1,3461	-0,2774	2,9695	0,6425	0,0363	0,6861	992,1882	0,0678	0,0654	0,9871

A Tabela 7.36 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Human com 40% de dados faltantes MAR. Novamente, o SwarmBcluster produziu resultados melhores que os do NORM e mais uma vez tanto a imputação utilizando a média como utilizando a mediana das 5 imputações de cada dado faltante produziram resultados melhores que os obtidos com a imputação única (Subseção 6.2.6). Porém, o ganho na qualidade das imputações é pequeno para justificar o maior custo computacional. O ganho no valor de MAE foi de 1,99% e o ganho no valor de RMSE foi de 2,69%.

Tabela 7.36: Resultados da estimação dos dados faltantes utilizando MI – Human com 40% de dados faltantes MAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
30,375	41,532	30,531	41,634	34,310	46,606	35,554	47,960	30,991	42,679

As Tabelas 7.37 e 7.38 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 80% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 18. Logo, o número de iterações de cada ciclo do DA foi igual a 20, totalizando 100 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster foi superior ao NORM, principalmente para os atributos 25, 34 e

40, como se pode observar através de r e γ. Tanto para o NORM quanto para a biclusterização, a média real dos atributos pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.37: Resultados da MI utilizando biclusterização – Human com 80% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
22	19,6810	17,0768	22,2853	1,7589	0,0054	1,7654	292936,7718	0,0037	0,0037	0,9993
25	5,4717	2,9385	8,0048	1,6420	0,0236	1,6704	13890,3474	0,0173	0,0171	0,9966
34	2,1325	0,1330	4,1319	1,0288	0,0099	1,0407	30937,2193	0,0115	0,0114	0,9977
40	2,5790	1,0294	4,1287	0,5586	0,0554	0,6251	353,0769	0,1191	0,1115	0,9782

Tabela 7.38: Resultados da MI utilizando NORM – Human com 80% de dados faltantes MAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
22	19,2979	16,4424	22,1534	1,9700	0,1271	2,1225	774,7939	0,0774	0,0742	0,9854
25	6,8465	3,7615	9,9314	1,8827	0,4955	2,4773	69,4364	0,3158	0,2610	0,9504
34	0,7468	-1,6625	3,1561	1,1755	0,2796	1,5110	81,1363	0,2854	0,2405	0,9541
40	1,5361	-0,2346	3,3068	0,6519	0,1369	0,8162	98,7218	0,2520	0,2170	0,9584

A Tabela 7.39 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Human com 80% de dados faltantes MAR. O SwarmBcluster produziu novamente resultados melhores que os do NORM e melhores que os obtidos na Subseção 6.2.6. Porém, quando se compara os valores obtidos através da imputação pela média ou mediana das 5 imputações com os resultados obtidos na Subseção 6.2.6, percebe-se que o ganho foi pequeno. O ganho no valor de MAE foi de apenas 1,32% e o ganho no valor de RMSE foi de apenas 2,26%.

Tabela 7.39: Resultados da estimação dos dados faltantes utilizando MI – Human com 80% de dados faltantes MAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
31,982	43,533	32,242	43,932	35,510	47,873	37,328	50,190	32,410	44,539

7.2.3 Dados faltantes segundo o mecanismo MNAR

As bases de dados com 40% e 80% de dados faltantes, segundo o mecanismo MNAR, utilizadas na Subseção 6.2.7 foram utilizadas neste experimento. Como descrito na Subseção

6.2.7, esses conjuntos de dados possuem dados faltantes no atributo 34. Os parâmetros adotados para a execução do SwarmBcluster foram os mesmos utilizados na Subseção 6.2.7.

As Tabelas 7.40 e 7.41 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 40% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 10. Logo, o número de iterações de cada ciclo do DA foi igual a 15, totalizando 75 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster apresentou imputações mais estáveis, como verifica-se através de r e γ. Entretanto, nem para o SwarmBcluster e nem o para NORM, a média real do atributo 34 pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.40: Resultados da MI utilizando biclusterização – Human com 40% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
34	-3,3081	-5,2855	-1,3307	1,0119	0,0049	1,0178	119581,4592	0,0058	0,0058	0,9988

Tabela 7.41: Resultados da MI utilizando NORM – Human com 40% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
34	-1,8716	-3,9808	0,2375	1,0864	0,0596	1,1580	1046,8015	0,0659	0,0636	0,9874

A Tabela 7.42 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Human com 40% de dados faltantes MNAR. Dessa vez, o NORM produziu resultados melhores que o SwarmBcluster. Mas, novamente, tanto a imputação pela média como pela mediana das 5 imputações de cada dado faltante produziram resultados melhores que os obtidos com a imputação única na Subseção 6.1.7. Porém, o ganho na qualidade das imputações é pequeno para justificar o maior custo computacional. O ganho no valor de MAE foi de apenas 0,87% e o ganho no valor de RMSE foi de 2,47%.

Tabela 7.42: Resultados da estimação dos dados faltantes utilizando MI – Human com 40% de dados faltantes MNAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
44,588	59,475	44,828	59,978	34,833	47,389	36,390	48,922	44,978	60,979

As Tabelas 7.43 e 7.44 trazem os resultados obtidos pela biclusterização e pelo NORM, respectivamente, para a base de dados com 80% de dados faltantes. O número de imputações foi igual a 5. O algoritmo EM convergiu na iteração 20. Logo, o número de iterações de cada ciclo do DA foi igual a 25, totalizando 125 iterações para produzir os cinco conjuntos de dados completos. O SwarmBcluster produziu estimativas mais estáveis que o NORM, como verifica-se através de r e γ. Ele também alcançou uma eficiência maior que o NORM. Mais uma vez, nem para o SwarmBcluster e nem o para NORM, a média real do atributo 34 pertence ao intervalo estabelecido por uma confiança de 95%.

Tabela 7.43: Resultados da MI utilizando biclusterização – Human com 80% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*R*	γ	*Efic.*
34	-10,2802	-11,9788	-8,5817	0,7410	0,0083	0,7510	22650,0165	0,0135	0,0134	0,9973

Tabela 7.44: Resultados da MI utilizando NORM – Human com 80% de dados faltantes MNAR (5 imputações).

Atr	*Média*	*95% Conf.*		$\overline{U}$	*B*	*T*	*df*	*r*	γ	*Efic.*
34	-6,3632	-8,2762	-4,4502	0,9099	0,0356	0,9526	1989,7716	0,0469	0,0458	0,9909

Tabela 7.45: Resultados da estimação dos dados faltantes utilizando MI – Human com 80% de dados faltantes MNAR (5 imputações).

Biclusterização				**NORM**					
Média		**Mediana**		**Média**		**Mediana**		**IU**	
MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
58,702	76,155	58,900	76,390	40,654	52,477	41,357	52,944	57,972	77,258

A Tabela 7.45 exibe os resultados da estimação dos dados faltantes utilizando MI para a base de dados Human com 80% de dados faltantes MNAR. Dessa vez, o NORM apresentou melhores resultados que o SwarmBcluster. E, pela primeira vez, o valor de MAE estimado pela imputação única foi melhor que o valor estimado através da média ou da mediana das 5 imputações. O valor de MAE aqui estimado é 1,26% pior que o obtido com a imputação única na Subseção 6.2.7. Já o valor de RMSE é 1,43% melhor que o obtido com a imputação única na Subseção 6.2.7. Isso indica que a imputação pela média ou mediana das 5 imputações

produziram, em média, estimativas piores que a imputação única, porém produziram erros absolutos máximos melhores.

7.3 Discussão

Os resultados obtidos pela biclusterização como modelo para a técnica de MI foram comparados com os obtidos pelo programa NORM (SCHAFER, 1999), um programa tradicional de MI. O NORM obteve ótimos resultados ao estimar o parâmetro de interesse definido neste trabalho, mais especificamente a média aritmética. Nesse aspecto, ele foi melhor que o SwarmBcluster, o que era esperado de um programa tradicional de MI, que foi desenvolvido para análises estáticas. Entretanto, as imputações realizadas pelo SwarmBcluster foram mais consistentes, de modo que na grande maioria das vezes ele produziu resultados melhores para as métricas definidas por Rubin, como o aumento relativo na variância devido aos dados faltantes (r) e a taxa de informação faltante (γ). É necessário, também, enfatizar que o NORM encontra bastante dificuldade para tratar bases de dados que possuem grandes quantidades de dados faltantes.

Nos testes que verificaram o quanto se poderia ganhar estimando um dado faltante através de imputações múltiplas em vez de uma imputação única, o SwarmBcluster foi consideravelmente melhor que o NORM. Salvo uma única exceção, esses testes também mostraram que essa técnica estima melhor os dados faltantes que a técnica de IU. Esses testes também mostraram que a imputação pela média das x imputações realizadas para cada dado faltante produziu, na grande maioria das vezes, resultados melhores que a imputação pela mediana. O grande problema é que o ganho em termos de qualidade de imputação é consideravelmente menor que o aumento no custo computacional. Mas também é importante frisar que as $x \geq 2$ execuções do algoritmo SwarmBcluster podem ser facilmente realizadas em paralelo, pois elas são independentes. Isso traria ganhos na qualidade da imputação e não traria nenhum custo computacional adicional. Outra característica importante a ser comentada é que os valores de MAE e RMSE apresentaram uma forte correlação positiva com os resultados obtidos pela métrica descrita por RUBIN (1987) para avaliar a qualidade da MI: a taxa de informação faltante. Para a biclusterização, a correlação entre os valores de MAE / RMSE (calculados tanto através da média como da mediana) com as médias da taxa de informação faltante de cada

experimento foi maior que 0,99. Com exceção dos experimentos realizados com a base de dados Human e o mecanismo MNAR, essa correlação também foi maior que 0,99 para o NORM. Além do mais, em praticamente todos os testes em que a biclusterização produziu melhores resultados para a taxa de informação faltante, também produziu melhores valores de MAE e RMSE que o NORM. Esses resultados podem ser úteis em aplicações como os sistemas de recomendação, em que não é possível avaliar a qualidade das sugestões através do valor de MAE ou RMSE, pois o valor real é desconhecido. Logo, essa métrica, que originalmente foi construída para avaliar a qualidade da MI, pode ser aplicada também para estimar a qualidade das sugestões de um sistema de recomendação.

A possível razão para que a biclusterização produza resultados superiores ao NORM é a mesma dos experimentos de imputação única: essa técnica cria vários modelos locais da base de dados para encontrar o subconjunto de objetos e atributos mais adequados para imputar um subconjunto de dados faltantes. Aplicando a biclusterização, através do SwarmBcluster, múltiplas vezes, é possível obter múltiplos modelos locais para cada valor faltante. Através da otimização quadrática, os valores faltantes presentes nesses modelos são estimados com o objetivo de maximizar a coerência entre os seus elementos. Como esses modelos locais sofrem menos influência do ruído do conjunto de dados e tendem a selecionar os objetos e atributos mais informativos para a imputação dos dados faltantes, essa pode ser a razão da biclusterização produzir resultados melhores. Esse cuidado ao estimar os valores faltantes reflete em pequenos valores de r (aumento relativo na variância devido aos dados faltantes), ou seja, reflete em menor variabilidade nos parâmetros de interesse estimados, combinada com uma confiabilidade adequada dessas estimativas. Logicamente, quanto menor o valor de r, melhores serão os valores de: graus de liberdade (*df*), taxa de informação faltante (γ) e eficiência (*Efic.*).

Novamente, o mecanismo MNAR foi o mais difícil de tratar. Para os experimentos realizados com a base de dados Yeast, os resultados foram bons tanto para o SwarmBcluster quanto para o NORM. Porém, o mesmo não aconteceu com os experimentos realizados com a base de dados Human. Ambos, SwarmBcluster e NORM, não foram capazes de estimar intervalos de confiança adequados para a média real dos atributos e, além disso, os resultados de MAE e RMSE foram consideravelmente ruins. Com relação ao SwarmBcluster, essa diferença de resultados entre as bases de dados Yeast e Human talvez seja explicada pelos resultados obtidos por DE FRANÇA & VON ZUBEN (2011), que mostraram que (*i*) a maioria dos biclusters

encontrados na base de dados Yeast possuem coerência aditiva (a qual foi utilizada neste trabalho); e (*ii*) a maioria dos biclusters encontrados na base de dados Human possuem coerência multiplicativa. Em trabalhos futuros, a imputação considerando a coerência multiplicativa pode ser avaliada, e, provavelmente, resultados melhores para a base de dados Human serão obtidos.

7.4 Síntese do capítulo

Este capítulo apresentou os experimentos realizados utilizando a biclusterização como modelo para a técnica de MI. Também mostrou o quanto se pode ganhar na qualidade da imputação quando um dado faltante é estimado com base em imputações múltiplas em vez de uma imputação única. Os resultados obtidos pela biclusterização, em ambos os experimentos, foram comparados com os obtidos pelo programa NORM (SCHAFER, 1999), um programa tradicional de MI.

Ao se trabalhar com a técnica de MI, o parâmetro de interesse escolhido foi a média aritmética. Na estimação de intervalos de confiança para a média, o NORM foi melhor sucedido que o SwarmBcluster. Porém, as imputações realizadas pelo SwarmBcluster foram mais consistentes, o que refletiu em melhores resultados nas métricas definidas por Rubin, como a taxa de informação faltante (γ).

Tanto para o SwarmBcluster quanto para o NORM, foi realizada a imputação da média, ou da mediana, das $x \geq 2$ estimativas de cada dado faltante obtidas por eles. O SwarmBcluster foi consideravelmente melhor que o NORM nesse quesito. Ademais, os resultados obtidos pelo SwarmBcluster com esse procedimento foram melhores que os resultados obtidos por ele com a IU. Ao se comparar os resultados obtidos pela média e pela mediana, a média, na grande maioria das vezes, produziu resultados melhores que a imputação pela mediana.

Mais uma vantagem da biclusterização com relação ao NORM é que ela conseguiu tratar todas as bases de dados. Já o NORM, encontrou bastante dificuldade para tratar bases de dados que possuíam grandes quantidades de dados faltantes.

Capítulo 8

Conclusão

Esta dissertação teve como objetivo principal a utilização da biclusterização conjuntamente com a imputação múltipla, de modo a (*i*) investigar o comportamento da biclusterização sendo utilizada como modelo para a imputação múltipla; (*ii*) avaliar o quanto se pode ganhar na qualidade da imputação quando um dado faltante é estimado com base em imputações múltiplas em vez de uma imputação única; e (*iii*) indicar como as métricas descritas por RUBIN (1987) para avaliar a qualidade da imputação múltipla podem ajudar a medir a qualidade da estimação dos valores faltantes.

O algoritmo de biclusterização utilizado foi o SwarmBcluster (DE FRANÇA & VON ZUBEN, 2010), o qual foi adaptado para trabalhar com a imputação de dados faltantes (DE FRANÇA, 2010). Por se tratar de uma proposta recente, esta dissertação também investigou esse algoritmo com os objetivos de (*i*) indicar como escolher os parâmetros corretamente e (*ii*) mostrar como esse algoritmo é competitivo para tratar dados faltantes de qualquer natureza.

O Capítulo 2 introduziu os principais conceitos relacionados aos dados faltantes. Foram apresentadas três características importantes relacionadas aos dados faltantes, que auxiliam na escolha de um método adequado para o tratamento dos dados faltantes. A primeira dessas características é o mecanismo associado aos dados faltantes, o qual está relacionado com a dependência entre valores de diferentes atributos, podendo ser: independente de qualquer evento (MCAR), dependente de outros atributos (MAR) ou dependente do próprio atributo (MNAR). A segunda dessas características é o padrão. O padrão indica se os dados faltantes ocorrem de forma não-estruturada ou sistemática. A terceira e última dessas características é a quantidade de dados faltantes. Esse capítulo também apresentou um caso particular que envolve dados faltantes: os sistemas de recomendação.

O Capítulo 3 descreveu alguns dos métodos mais conhecidos para o tratamento de dados faltantes, como também as vantagens e desvantagens de cada um desses métodos. Basicamente, os métodos apresentados podem ser divididos em dois grupos: (*i*) aqueles que almejam predizer

os parâmetros de interesse; e (*ii*) aqueles que almejam predizer os valores dos dados faltantes. Além disso, foram introduzidos os conceitos teóricos de Imputação Múltipla e como essa técnica é geralmente empregada na literatura.

O Capítulo 4 introduziu a técnica de biclusterização, a qual é capaz de extrair subconjuntos de linhas e colunas de uma matriz de dados, de modo que os elementos desses subconjuntos sejam, de alguma forma, similares. O critério utilizado neste trabalho para medir essa similaridade é a chamada *coerência aditiva* (CHENG & CHURCH, 2000).

Os três algoritmos que foram utilizados nos experimentos de imputação única foram descritos no Capítulo 5. O primeiro deles foi o algoritmo KNNImpute. Esse algoritmo é baseado no algoritmo de *k-Vizinhos mais próximos* para imputação de dados faltantes, descrito em CANDILLIER (2008). O segundo deles foi o rSVD, um algoritmo baseado em *Decomposição de Valores Singulares com Regularização*, criado por FUNK (2006). O terceiro algoritmo foi o SwarmBcluster, um algoritmo de biclusterização inspirado em Inteligência de Enxames, que foi adaptado para a imputação de dados faltantes por DE FRANÇA (2010). O algoritmo SwarmBcluster também foi utilizado nos experimentos envolvendo biclusterização e imputação múltipla.

No Capítulo 6 foram realizados experimentos de análise de sensibilidade paramétrica do algoritmo SwarmBcluster, assim como experimentos comparando o desempenho dos algoritmos apresentados anteriormente, SwarmBcluster, KNNImpute e rSVD, perante diferentes conjuntos de dados, mecanismos associados aos dados faltantes e quantidades de dados faltantes.

Por fim, o Capítulo 7 apresentou os resultados obtidos com os experimentos que trabalharam conjuntamente as técnicas de biclusterização e imputação múltipla. Esses experimentos também foram realizados com diferentes conjuntos de dados, mecanismos associados aos dados faltantes e quantidades de dados faltantes.

8.1 Discussão

Os experimentos de análise de sensibilidade paramétrica mostraram que o parâmetro que exerce maior influência na qualidade da imputação do SwarmBcluster é o δ, o qual controla diretamente a intensidade máxima de ruído presente no bicluster. O melhor valor de δ depende da quantidade de dados faltantes. Para quantidades pequenas ou médias, o melhor valor de δ tende a

ser menor do que para grandes quantidades. Isso porque, quando a base contém muitos dados faltantes, a intensidade do ruído, por conta da ausência de valores, é maior e, consequentemente, deve-se permitir que os biclusters encontrados tenham um ruído também maior.

A influência dos outros dois parâmetros testados, max_it e n_ants, foi pequena. Apenas para o teste realizado com o conjunto de dados com 80% de dados faltantes, criado a partir da base de dados Yeast, a influência de max_it foi significativa. Isso reforça que, para grandes quantidades de dados faltantes, é vantajoso beneficiar o volume do bicluster. Assim, a taxa de dados faltantes dentro do bicluster tende a ser menor, reduzindo a influência do ruído e tornando o problema de otimização correspondente factível.

Os testes realizados com a heurística construtiva (max_it = 1 e n_ants = 1) mostraram que é possível reduzir o custo computacional enquanto se obtém uma boa qualidade de imputação.

Os experimentos de comparação de desempenho para imputação única entre o SwarmBcluster, o KNNImpute e o rSVD mostraram que, na grande maioria das vezes, o SwarmBcluster possui desempenho consideravelmente melhor. A única desvantagem do SwarmBCluster foi o custo computacional.

A biclusterização também se mostrou uma estratégia eficaz e flexível para gerar as imputações múltiplas. Os resultados da biclusterização foram competitivos com os resultados obtidos pelo programa NORM, um programa tradicional de imputação múltipla e que foi desenvolvido com a finalidade de estimar parâmetros de interesse.

Os experimentos envolvendo biclusterização e imputação múltipla também mostraram que essa técnica obteve uma melhor estimativa dos dados faltantes em relação à técnica de imputação única, principalmente para conjuntos de dados com grande quantidade de dados faltantes. O aumento do custo computacional desse procedimento pode ser aliviado através de múltiplas execuções em paralelo do algoritmo de biclusterização, no caso, o SwarmBcluster.

Outro resultado bastante interessante é que os valores de MAE e RMSE apresentaram uma forte relação com os valores da taxa de informação faltante. Em situações reais, não é possível estimar a qualidade das imputações através de métricas que se baseiam no valor real, como no caso de MAE e RMSE. Logo, esse resultado abre uma nova possibilidade para avaliar a qualidade das imputações em problemas reais de dados faltantes, como os sistemas de recomendação.

8.2 Perspectivas Futuras

Com os estudos realizados nesta dissertação, é possível perceber vários aspectos a serem explorados nessa linha de pesquisa. Um deles, dando sequência à análise de sensibilidade. Poderia ser analisado se a combinação dos parâmetros testados influencia a qualidade final das imputações, para saber se existe alguma correlação significativa entre os parâmetros. Além disso, a análise realizada aqui pode ser realizada em outros conjuntos de dados para oferecer uma forma de estimar o melhor conjunto de parâmetros, dadas as características desse conjunto e o mecanismo responsável pela falta de dados. Adicionalmente, poderiam ser realizados experimentos avaliando o comportamento da biclusterização em problemas envolvendo os três mecanismos de ausência de dados conjuntamente. Também é possível analisar a influência dos outros parâmetros do SwarmBcluster na qualidade da imputação. E, por fim, poderiam ser avaliadas outras formas de coerência para a construção do biclusters, como a coerência multiplicativa.

As métricas adotadas neste trabalho para medir a qualidade da imputação, RMSE e MAE, são amplamente utilizadas. Entretanto, essas métricas consideram que o valor real da base de dados completa não é ruidoso, o que nem sempre é verdadeiro. Portanto, definir novas métricas, que não dependam do valor real, é um trabalho importante, tanto para problemas reais de dados faltantes, como para trabalhos acadêmicos que avaliam o desempenho de algoritmos.

Também podem ser estudadas outras estratégias para utilizar a biclusterização e a imputação múltipla conjuntamente, como a simples variação de δ para gerar as imputações múltiplas. Além disso, é possível estudar outras formas de agrupar as imputações múltiplas em uma única. É possível, por exemplo, ponderar cada uma das múltiplas imputações com base na qualidade do bicluster.

Para finalizar, poderá ser feita uma análise mais detalhada da relação entre o resíduo quadrático médio e o volume com o erro de estimação do valor faltante. Um estudo preliminar foi feito em DE FRANÇA (2010), porém isso deve ser atestado com novos experimentos e, também, colocado em prática em problemas reais que necessitam de tal informação, como, por exemplo, os sistemas de recomendação.

REFERÊNCIAS BIBLIOGRÁFICAS

E. ACUNA, C. RODRIGUEZ. The treatment of missing values and its effect in the classifier accuracy. *Classification, Clustering and Data Mining Applications*, vol. 0, pp. 639-648, 2004.

G. ADOMAVICIUS, A. TUZHILIN. 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*, vol. 17, no. 6, pp. 734-749, 2005.

R. AGRAWAL, J. E. GEHRKE, D. GUNOPULOS, P. RAGHAVAN. Automatic subspace clustering of high dimensional data for data mining applications. *Proceedings of the ACM/SIGMOD Int. Conference on Management of Data,* pp. 94–105, 1999.

A. A. ALIZADEH, M. B. EISEN, R. E. DAVIS, C. MA, I. S. LOSSOS, A. ROSENWALD, J. C. BOLDRICK, H. SABET, T. TRAN, X. YU, J. I. POWELL, L. YANG, G. E. MARTI, T. MOORE, J. HUDSON JR, L. LU, D. B. LEWIS, R. TIBSHIRANI, G. SHERLOCK, W. C. CHAN, T. C. GREINER, D. D. WEISENBURGER, J. O. ARMITAGE, R. WARNKE, R. LEVY, W. WILSON, M. R. GREVER, J. C. BYRD, D. BOTSTEIN, P. O. BROWN, L. M. STAUDT. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, vol. 403, pp. 503-511, 2000.

P. D. ALLISON. Estimation of linear models with incomplete data. *Sociological Methodology*, vol. 17, pp. 71-103, 1987.

P. D. ALLISON. *Missing data*. Sage University Papers Series on Quantitative Applications in the Social Sciences, 07-136. Thousand Oaks, CA: Sage, 2001.

J. BENNETT, S. LANNING. The Netflix Prize. *Proceedings of KDD Cup and Workshop*, San Jose, 2007.

P. BERKHIN. A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pp. 25-71, 2006.

C. BLUM, M. DORIGO. The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34 no. 2, pp. 1161-1172, 2004.

J. BREESE, D. HECKERMAN, C. KADIE. Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52. Morgan Kaufman, 1998.

G. N. BROCK, J. R. SHAFFER, R. E. BLAKESLEY, M. J. LOTZ, G. C. TSENG. Which missing value imputation method to use in expression profiles: a comparative study and two selection schemes. *BMC Bioinformatics*, vol. 12, pp. 1-12, 2008.

M. L. BROWN, J. F. KROS. Data mining and the impact of missing data. *Industrial Management & Data Systems***,** vol. 103, no. 8, pp. 611-621, 2003.

S. V. BUUREN, E. M. V. MULLIGEN, J. P. L. BRAND. Routine multiple imputation in statistical databases. *Proceedings of the International Working Conference On Seventh Scientific And Statistical Database Management*, pp.74-78, 1994.

L. CANDILLIER, K. JACK, F. FESSANT, F. MEYER. State-of-the-Art recommender systems. *Collaborative and Social Information Retrieval and Access – Techniques for Improved User Modeling*, pp. 1-22, 2008.

R. B. CATTEL. The data box: Its ordering of total resources in terms of possible relations systems. *Handbook of Multivariate Experimental Psychology*, pp. 67-128. Chicago: Rand McNally, 1996.

Y. CHENG, G. M. CHURCH. Biclustering of expression data. *Proceedings. of the 8th International Conference on Intelligent Systems for Molecular Biology*, pp. 93–103, 2000.

R. CHO, M. CAMPBELL, E. WINZELER, L. STEINMETZ, A. CONWAY, L. WODICKA, A. GABRIELIAN, D. LANDSMAN, D. LOCKHART, R. DAVIS. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, vol. 2, no. 1, pp. 65-73, 1998.

G. P. COELHO, F. O. DE FRANÇA, F. J. VON ZUBEN. Multi-Objective Biclustering: When Non-dominated Solutions are not Enough. *Journal of Mathematical Modelling and Algorithms*, vol. 8, pp. 175-202, 2009.

A. COLANTONIO, R. D. PIETRO, A. OCELLO, N. V. VERDE. ABBA: Adaptive Bicluster-Based approach to impute missing values in binary matrices. *Proceedings of the 2010 ACM Symposium on Applied Computing*, pp. 1026-1033, 2010.

L. N. DE CASTRO. *Fundamentals of natural computing: basic concepts, Algorithms, and Applications*. Chapman & Hall/CRC, 2006.

P. A. D. DE CASTRO, F. O. DE FRANÇA, H. M. FERREIRA, G. P. COELHO, F. J. VON ZUBEN. Query expansion using an immune-inspired biclustering algorithm. *Natural Computing*, vol. 9, no. 3, pp. 579-602, 2007a.

P. A. D. DE CASTRO, F. O. DE FRANÇA, H. M. FERREIRA, F. J. VON ZUBEN. Applying biclustering to perform collaborative filtering. *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications*, Rio de Janeiro, pp. 421-426, 2007b.

P. A. D. DE CASTRO, F. O. DE FRANÇA, H. M. FERREIRA, F. J. VON ZUBEN. Applying biclustering to text mining: an immune-inspired approach. *Artificial Immune Systems, Proc. of the 6th International Conference on Artificial Immune Systems*, Santos, Lectures Notes in Computer Science, vol. 4628 pp. 83-94, 2007c.

P. A. D. DE CASTRO, F. O. DE FRANÇA, H. M. FERREIRA, F. J. VON ZUBEN. Evaluating the performance of a biclustering algorithm applied to collaborative filtering: a comparative analysis. *Proceedings. of the 7th International Conference on Hybrid Intelligent Systems*, Kaiserslautern, pp. 65–70, 2007d.

F. O. DE FRANÇA, F. J. VON ZUBEN, L. N. DE CASTRO. A max min ant system applied to the capacitated clustering problem. *Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*, pp. 755–764, 2004a.

F. O. DE FRANÇA, F. J. VON ZUBEN, L. N. DE CASTRO. Definition of capacited p-medians by a modified max min ant system with local search. *Proceedings of the Neural Information Processing*, vol. 3316, pp. 1094–1100. Springer, 2004b.

F. O. DE FRANÇA, F. J. VON ZUBEN, L. N. DE CASTRO. Max min ant system and capacitated p-medians: extensions and improved solutions. *Informatica*, vol, 29, pp. 163–171, 2005.

F. O. DE FRANÇA, G. BEZERRA, F. J. VON ZUBEN. New perspectives for the biclustering problem. *Proceedings of the 12th IEEE Congress on Evolutionary Computation*, Vancouver, pp. 753-760, 2006.

F. O. DE FRANÇA, F. J. VON ZUBEN. Finding a high coverage set of δ-biclusters with swarm intelligence. *Proceedings of the 12th IEEE Congress on Evolutionary Computation*, pp. 2523-2530, 2010.

F. O. DE FRANÇA. *Biclusterização na análise de dados incertos*. Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas, 2010.

F. O. DE FRANÇA, F. J. VON ZUBEN. Extracting Additive and Multiplicative Coherent Biclusters with Swarm Intelligence. *IEEE Congress on Evolutionary Computation*, pp. 632-638, 2011.

K. DEB, A. PRATAP, S. AGARWAL, T. MEYARIVAN. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

M. DORIGO. *Optimization, learning and natural algorithms*. Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

A. FARHANGFAR, L. KURGAN, W. PEDRYCZ. Experimental analysis of methods for imputation of missing values in databases. *Proceedings of SPIE*, Orlando, vol. 5421, pp. 172-182, 2004.

A. FARHANGFAR, L. KURGAN, W. PEDRYCZ. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 5, pp. 692-709, 2007.

R. FELDMAN, J. SANGER. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge, MA: Cambridge University Press, 2007.

J. H. FRIEDMAN, J. BENTELY, R. A. FINKEL. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, vol. 3, pp. 209, 1977.

S. FUNK. (2006). Netflix update. Retrieved from *http://sifter.org/~simon/journal/20061211.html*

S. FUNK. (2007) Netflix update. Retrieved from *http://sifter.org/~simon/journal/20070815.html*

D. GOLDFARB, A. IDNANI. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, vol. 27, pp. 1-33, 1983.

K. GOLDBERG, T. ROEDER, D. GUPTA, C. PERKINS. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, vol. 4, no. 2, pp. 133-151, 2001.

S. GOSS, S. ARON, J. DENEUBOURG, J. PASTEELS. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, vol. 76, no. 12, pp. 579-581, 1989.

J. W. GRAHAM, S. M. HOFER, S. I. DONALDSON, D. P. MACKINNON, J. L. SCHAFER. Analysis with missing data in prevention research. *The science of prevention: Methodological advances from alcohol and substance abuse research*, Washington, pp. 325–366, 1995.

J. W. GRAHAM. Missing data analysis: making it work in the real world. *The Annual Review Of Psychology*, Pennsylvania, pp. 549-576, 2009.

J. A. HARTIGAN. Direct clustering of a data matrix. *Journal of the American Statistical Association*, vol. 67 no. 337, pp. 123-129, 1972.

T. HASTIE, R. TIBSHIRANI, M. EISEN, P. BROWN, G. SHERLOCK, D. BOTSTEIN. Imputing missing data for gene expression arrays. *Stanford Statistics Department,* Stanford, 1999.

S. L. HERSHBERGER, D. G. FISHER. A note on determining the number of imputations for missing data. *Structural Equation Modeling*, vol. 10, pp. 648-550, 2003.

P. JACCARD. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

C. M. JARQUE, A. K. BERA. A test for normality of observations and regression residuals. *International Statistical Review*, vol. 55, no. 2, pp. 163–172, 1987.

J. KENNEDY, R. EBERHART, Y. SHI. *Swarm intelligence*. Springer, 2001.

Y. KOREN, R. BELL. *Advances in Collaborative Filtering*: Recommender Systems Handbook. US: Springer, 2011.

J. KRUSKAL JR. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48-50, 1956.

K. LAKSHMINARAYAN, S. A. HARP, T. SAMAD. Imputation of missing data in industrial databases. *Applied Intelligence***,** Minneapolis, pp. 259-275, 1999.

H. W. LILLIEFORS. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*. vol. 62, pp. 399–402, 1967.

W. LIN, S. ALVAREZ, C. RUIZ. Efficient adaptive-support association rule mining for recommender systems. *In Data Mining and Knowledge Discovery*, vol. 6, pp. 83-105, 2002.

R. J. A. LITTLE. A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association*, vol. 83, pp. 1198-1202, 1988.

R. J. A. LITTLE, D. B. RUBIN. *Statistical analysis with missing data***,** 2. ed. Hoboken: John Wiley & Sons, 2002.

S. C. MADEIRA, A. L. OLIVEIRA. Biclustering algorithms for biological data analysis: a survey. *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24-45, 2004.

F. J. MASSEY. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*. vol. 46, no. 253, pp. 68–78, 1951.

P. MELVILLE, R. MOONEY, R. NAGARAJAN. Content-boosted collaborative filtering for improved recommendations. *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 187-192, 2002.

P. E. MCKNIGHT, K. M. MCKNIGHT, S. SIDANI, A. J. FIGUEREDO. *Missing data: a gentle introduction*. New York: The Guilford Press, 2007.

X. L. MENG, D. B. RUBIN. Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika*, vol. 80, pp. 267–278, 1993.

B. MIRKIN. *Mathematical Classification and Clustering*. Springer, 1996.

S. MITRA, H. BANKA. Multi-objective evolutionary biclustering of gene expression data. *pattern recognition*, vol. 39, no. 12, pp. 2464–2477, 2006.

A. MOHAMMADI, M. H. SARAEE. Dealing with missing values in microarray data. *Proceedings of the 4th International Conference on Emerging* Technologies, Rawalpindi, pp. 258-263, 2008.

B. MUTHÉN, D. KAPLAN, M. HOLLIS. On structural equation modeling with data that are not missing completely at random. *Jornal of Psychometrika*, Springer New York, vol. 52, no. 3, pp. 431-462, 1987.

I. MYRTVEIT, E. STENSRUD, U. H. OLSSON. Analyzing data sets with missing data: an empirical evaluation of imputation methods and likelihood-based methods. *IEEE Transactions On Software Engineering*, vol. 27, no. 11, pp. 999-1013, 2001.

M. O'CONNER, J. HERLOCKER. Clustering items for collaborative filtering. *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

A. PATEREK. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, San Jose, 2007.

G. POLCICOVA, R. SLOVAK, P. NAVRAT. Combining content-based and collaborative filtering. *Proceedings of the ADBIS-DASFAA Symposium*, pp. 118-127, 2000.

D. B. RUBIN. *Multiple imputation for nonresponse in surveys*. New York: Wiley, 1987.

B. M. SARWAR, G. KARYPIS, J. KONSTAN, J. RIEDL. Analysis of recommendation algorithms for e-commerce. *Proceedings of the ACM Conference on Electronic Commerce*, pp. 158-167, 2000.

J. L. SCHAFER. *Analysis of incomplete multivariate data*. Londom: Chapman & Hall, 1997.

J. L. SCHAFER. (1999) *NORM*: *Multiple imputation of incomplete multivariate data under a normal model*, version 2. Software for Windows 95/98/NT, available from http://www.stat.psu.edu/~jls/misoftwa.html.

J. L. SCHAFER, J. W. GRAHAM. Missing data: our view of the state of the art. *Psychological Methods*, vol. 7, no. 2, pp. 147-177, 2002.

S. STIGLER. Francis Galton's account of the invention of correlation. *Statistical Science*, vol. 4, no. 2, pp. 73–79, 1989.

T. STÜTZLE, H. H. HOOS. MAX-MIN ant system. *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889-914, 2000.

P. SYMEONIDIS, A. NANOPOULOS, A. PAPADOPOULOS, Y. MANOLOPOULOS. Nearest-biclusters collaborative filtering with constant values. *Advances in Web Mining and Web Usage Analysis*, Philadelphia, vol. 4811, pp. 36-55, 2007.

A. TANAY, R. SHARAN, R. SHAMIR. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, vol. 18, pp. 136-144, 2002.

M. A. TANNER, W. H. WONG. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 528-540, 1987.

S. TAVAZOIE, J. HUGHES, M. CAMPBELL, R. CHO, G. CHURCH. Systematic determination of genetic network architecture. *Nature Genetics*, vol. 22, pp. 281-285, 1999.

M. F. TRIOLA. *Introdução à estatística*. 9. ed. Rio de Janeiro: Ltc, 2005.

O. TROYANSKAYA, M. CANTOR, G. SHERLOCK, P. BROWN, T. HASTIE, R. TIBSHIRANI, D. BOTSTEIN, R. B. ALTMAN. Missing value estimation methods for DNA microarrays. *Bioinformatics*, vol. 17, no. 6, pp. 520-525, 2001.

L. UNGAR, D. FOSTER. Clustering methods for collaborative filtering. *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, 1998.

C. WU, C. WUN, H. CHOU. Using association rules for completing missing data. *Proceedings of the 4th International Conference On Hybrid Intelligent Systems*, Taiwan, pp. 236-241, 2004.

P. ZHANG. Multiple imputation: theory and method. *International Statistical Review*, vol. 71, no. 3, pp. 581-592, 2003.

C. ZHANG, Y. QIN, X. ZHU, J. ZHANG, S. ZHANG. Clustering-based missing value imputation for data preprocessing. *Industrial Informatics*, pp. 1081-1086, 2006.

Q. ZHAO, P. MITRA, D. LEE, J. KANG. HICCUP: hierarchical clustering based value imputation using heterogeneous gene expression microarray datasets. *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering*, Boston, pp. 71-78, 2007.

Trabalho publicado pela autora

R. VERONEZE, F. O. de FRANÇA, F. J. VON ZUBEN. "Assessing the performance of a swarm-based biclustering technique for data imputation." In: *IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, pp. 386-393, 2011.