UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

TRATAMENTO DE VALORES AUSENTES EM SÉRIES TEMPORAIS MULTIVARIADAS POR MEIO DE PROGRAMAÇÃO GENÉTICA

DAMARES CRYSTINA OLIVEIRA DE RESENDE

BELÉM - PARÁ
2016

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

TRATAMENTO DE VALORES AUSENTES EM SÉRIES TEMPORAIS MULTIVARIADAS POR MEIO DE PROGRAMAÇÃO GENÉTICA

DAMARES CRYSTINA OLIVEIRA DE RESENDE

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Engenheiro em Engenharia da Computação, do Instituto de Tecnologia, da Faculdade de Engenharia da Computação e Telecomunicações.

BELÉM - PARÁ
2016

TRATAMENTO DE VALORES AUSENTES EM SÉRIES TEMPORAIS MULTIVARIADAS POR MEIO DE PROGRAMAÇÃO GENÉTICA

Este trabalho foi julgado adequado em ____/____/_____ para a obtenção do Grau de Engenheiro da Computação, aprovado em sua forma final pela banca examinadora que atribuiu o conceito _____.

Prof. Dr. Fábio Manoel França Lobato
Instituto de Engenharia e Geociências - UFOPA
ORIENTADOR

Prof. Dr. Ádamo Lima de Santana
Faculdade de Engenharia da Computação e Telecomunicações - UFPA
COORIENTADOR

Prof. Dr. Claudomiro de Souza de Sales Júnior
Faculdade de Computação - UFPA
MEMBRO DA BANCA EXAMINADORA

Prof. MSc. Antônio Fernando Lavareda Jacob Júnior
Centro de Ciências Tecnológicas - UEMA
MEMBRO DA BANCA EXAMINADORA

VISTO:

Prof. Dr. Francisco Carlos Bentes Frey Müller
DIRETOR DA FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

BELÉM - PARÁ

2016

"Quem come do fruto do conhecimento é sempre expulso de algum paraíso."

Melanie Klein

À meus pais e irmão, os pilares de minha vida.

Agradecimentos

Agradeço primeiramente a minha mãe Maria Orlandina, ao meu pai Sebastião e ao meu irmão Carlos por todo o suporte, carinho, atenção, força e amor e por sempre acreditarem em mim e me incentivarem a correr atrás dos meus sonhos.

Agradeço a todos meus amigos de faculdade que comigo estudaram, trabalharam, riram, "choraram" e se divertiram, tornando estes cinco anos inesquecíveis. Agradeço em especial a meus amigos Ana Larissa e Willian, que estiveram ao meu lado tanto em momentos alegres quanto nos momentos mais difíceis.

Agradeço aos meus amigos de intercâmbio que comigo exploraram um novo universo e viveram experiências incríveis, expandindo minha mente para a diversidade e novos desafios.

Agradeço aos meus professores que tanto me ensinaram e que me fizeram enxergar o mundo como um grande espaço a ser continuamente explorado. Agradeço também a todos aqueles que contribuíram direta e indiretamente para minha formação.

Agradeço ao meu orientador Fábio Lobato por ter acreditado em mim e por toda a paciência e dedicação ao me orientar nas pesquisas que culminaram neste trabalho.

Agradeço a todos do Laboratório de Inteligência Computacional e Pesquisa Operacional (LINC) e do Laboratório de Processamento de Sinais (LaPS) que comigo pesquisaram e aprenderam, e que me ensinaram as diretrizes do trabalho acadêmico, profissional e em equipe.

Agradeço a Universidade Federal do Pará, a Stony Brook University e a State University of New York at New Paltz por me proporcionarem uma experiência acadêmica que me permitiu superar desafios e ampliar meus horizontes, me fazendo amadurecer e crescer interpessoalmente e profissionalmente.

Damares Crystina Oliveira de Resende

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API - *Application Programming Interface*

AR - *Autoregressive Model*

ARMA - *Mixed Autoregressive-Moving Average Models*

ARIMA - *Autoregressive Integrated Moving Average Model*

ECJ - *Evolutionary Computation Toolkit*

EM - *Exact Match*

KDD - *Knowledge-Discovery in Databases*

KNNI - *k Nearest Neighbor Imputation*

MA - *Moving Average Model*

MAR - *Missing at random*

NMAR - *Not missing at random*

MCAR - *Missing completely at random*

PG - Programação Genética

PSO - *Particle Swarm Optimization*

SVR - *Support Vector Regression*

VA - Valores Ausentes

Resumo

Um problema ubíquo inerente à análise de dados é a ocorrência de valores ausentes na base, o que provoca um viés no conjunto de informações quando não tratados corretamente. Este problema é ainda mais complexo no âmbito de séries temporais, pois a ausência de algumas amostras corrompe a estrutura temporal e afeta medidas estatísticas que caracterizam a distribuição dos dados. Visando mitigar este problema, neste trabalho são projetados e implementados dois algoritmos para tratamento de valores ausentes em séries temporais multivariadas. Ambos fazem uso de programação genética baseada em uma função de aptidão multicritério e objetivam estimar os dados faltosos na base, mantendo as características estatísticas originais da distribuição. Os algoritmos desenvolvidos se mostraram promissores: primeiramente, diferentemente de grande parte dos métodos presentes na literatura, os algoritmos constroem um modelo compreensível que pode ser usado por analistas para o estudo de padrões no conjunto de dados; ademais, no caso do segundo método proposto, além de manter as propriedades estatísticas das séries temporais, ele é também um bom mecanismo para imputar dados em bases desta natureza.

Palavras-chave: Séries Temporais, Programação Genética, Imputação

Abstract

An ubiquitous problem inherent from data analysis is the presence of gaps in the database which, when not treated correctly, introduces bias on the dataset. This problem is even more complex when dealing with time series, since the absence of data corrupts the temporal structure, affecting statistical measures that characterize the data distribution. Aiming to mitigate this issue, in this project it is planned and implemented two algorithms to treat missing values in multivariate time series data. Both make use of genetic programing based on a multi-criteria fitness function, and intent to estimate missing values in the dataset keeping the original statistical properties of the distribution. The algorithms developed are promising: firstly, differently of great part of the methods in the literature, the algorithms build a comprehensive model that can be used by data analysts to study the dataset's patterns; moreover, considering the second algorithm proposed, in addition to keep the statistical properties of a time series, it is also a great mechanism to impute data in this type of databases.

Keywords: Time Series, Genetic Programming, Imputation

Sumário

1. Introdução

Séries temporais são definidas como um processo estocástico proveniente da coleta de uma sequência de amostras ao longo do tempo (BOX, JENKINS, REINSEL, 2008). Estas estruturas estão presentes em diversas áreas de aplicação: a exemplo da meteorologia, onde é feito o monitoramento das condições do ambiente como temperatura e umidade; economia, que faz uma análise da bolsa de valores em uma determinada época sob determinadas condições; medicina, que monitora sinais biológicos e fisiológicos de um paciente, como pressão sanguínea e batimentos cardíacos; ou mesmo sociologia e política, áreas que avaliam estatísticas sociais e econômicas de uma região, como densidade demográfica e taxas de desemprego.

A partir da análise de séries temporais é possível identificar alguns padrões de comportamento, e desta forma pode-se fazer predições de eventos ou identificar falhas e parâmetros de sensibilidade em um processo (SHUMWAY, STOFFER, 2010). Na área de meteorologia, por exemplo, pode-se dizer se irá chover amanhã a partir da análise da temperatura e umidade hoje, de forma que amostras em um tempo $t - z$ poderão indicar o valor de amostras em um tempo t. Em economia, pode-se dizer se é viável fazer uma aplicação em um país a partir do estudo de sua moeda, em conjunto com uma análise de mercado e da variação do crescimento da bolsa de valores. Já no âmbito da medicina, pode-se identificar fatores agravantes de uma doença a partir do estudo da resposta do paciente a determinadas situações (SHUMWAY, STOFFER, 2010), (JUNGER, LEON, 2015), (BRUNO et al., 2013) e (HONAKER, KING, 2010).

A estrutura de série temporal possui características específicas advindas da relação entre os dados de cada amostra com as amostras vizinhas, apresentando também variações sutis entre um valor e outro, assim como periodicidade ou sazonalidade ao longo do tempo (BOX, JENKINS, REINSEL, 2008). Devido a estas particularidades, séries temporais são tratadas de maneira ímpar, sendo necessárias adaptações nos modelos de análise e predição.

Um infortúnio comumente enfrentado na análise de dados é a presença de lacunas no conjunto de informações estudadas. A ausência de valores em uma base é um problema ubíquo

proveniente de complicações enfrentadas na etapa de aquisição e seleção de dados (LOBATO et al., 2015). Este fenômeno é comumente provocado por falhas nos sensores responsáveis pela leitura e armazenamento, mudanças no modelo do experimento durante a aquisição, ou mesmo o não preenchimento de algumas respostas em pesquisas, por exemplo (LITTLE, RUBIN, 1987). Como consequência, as lacunas presentes na distribuição das amostras prejudicam o estudo feito sobre esta informação, pois a maioria dos modelos lineares de análise exigem a completude dos dados. Ademais, a ausência de valores na base gera um viés na estrutura, prejudicando assim a extração de conhecimento deste conjunto de dados e conduzindo o estudo a resultados inconsistentes (LUENGO, GARCÍA, HERRERA, 2012) e (HRUSCHKA et al., 2009).

Em séries temporais este problema é ainda mais complexo, pois as lacunas deixadas por dados faltosos corrompem a estrutura temporal, afetando valores estatísticos que caracterizam estes dados, tais como a auto-correlação das amostras, as tendências da série e também sua sazonalidade (HONAKER, KING, 2010). Por este motivo, é vital que os valores ausentes sejam tratados antes que seja feito um estudo mais aprofundado sobre o conjunto de dados.

Neste contexto, este trabalho propõe dois métodos de tratamento de valores ausentes em séries temporais multivariadas, tendo como base Programação Genética (PG) para gerar funções de regressão que estimem os dados faltosos. Segundo Tran, Zhang e Andreae (2015), este Algoritmo Evolucionário (AE) é capaz de aprender a definição de funções a partir de exemplos de dados e portanto é uma escolha interessante como algoritmo de regressão, sendo também uma opção promissora para aplicação em imputação de dados. É baseado nesta hipótese que esta abordagem foi escolhida para ser implementada no projeto.

No presente trabalho são projetados, codificados e testados dois algoritmos de imputação de dados, chamados de GPImpute e LGPImpute. Os algoritmos desenvolvidos tem por objetivo tratar valores ausentes em séries temporais e manter as características estatísticas originais de sua distribuição, como média, variância e autocorrelação, de forma a reduzir o viés imposto pela presença de lacunas nos dados. Ambos métodos fazem uso de PG para criar funções de regressão a partir da análise das interrelações de séries temporais multivariadas, e estimam por meio delas os valores ausentes na estrutura. Esta PG é codificadas a partir do *framework Evolutionary Computation Toolkit* (ECJ) (LUKE etal., 2004), ferramenta escrita em Java que oferece a implementação de diversos AE.

As bases de dados utilizadas para testar os algoritmos desenvolvidos são sintéticas, de forma que 5%, 10% e 30% das amostras foram amputadas em cada base. Além disso, duas

métricas de avaliação são aplicadas para indicar a eficiência de cada método: RMSE, que indica o quão distante o valor estimado é do valor real; e a diferença de autocorrelação da distribuição, que indica a capacidade do algoritmo de manter as propriedades originais da série temporal após a imputação dos dados. Por fim, em ambos algoritmos desenvolvidos, foram executados 10 testes para cada uma das 21 bases estudadas.

Apesar de serem similares, estes algoritmos possuem diferenças importantes na forma como analisam os dados. Para construir os modelos de regressão, devido o fato deles usarem valores dos atributos vizinhos para gerar as funções, é necessário fazer uma pré-imputação na base. O GPImpute usa os valores de amostras adjacentes as lacunas para substituir os valores ausentes, enquanto LGPImpute o faz por meio da interpolação de Lagrange. Após este processo, executa-se a programação genética que gerará as funções de regressão, onde apenas no caso do LGPImpute, amostras passadas são consideradas para construí-las.

A maior vantagem inerente ao uso destas abordagens é o fato de ambas gerarem um modelo regressivo compreensível, de forma que são construídas equações polinomiais que definem um padrão para a distribuição dos valores ausentes em cada atributo da série temporal. Além disso, o GPImpute e LGPImpute se mostram eficientes para manter os valores estatísticos originais da distribuição dos dados. Ademais, no caso do LGPImpute, a acurácia de imputação é competitiva em relação aos métodos EC e MC em todos os conjuntos de bases analisadas, e é competitiva em relação ao KMI quando considerados os melhores resultados das 10 execuções de teste feitas.

Este trabalho está organizado da seguinte forma: O Capítulo 2 aborda alguns dos trabalhos mais relevantes que inspiraram o método proposto. O Capítulo 3 alude a base teórica fundamental para a implementação do algoritmo, que envolve conceitos de Análise de Dados, Séries Temporais, Programação Genética e Interpolação de Lagrange. O Capítulo 4 retrata detalhes da implementação do GPImpute e do LGPImpute. O Capítulo 5 trata dos experimentos realizados e resultados obtidos a partir do estudo feito, e por fim, o Capítulo 6 apresenta as conclusões do trabalho.

2. Trabalhos Correlatos

Como dito anteriormente, a ausência de valores ausentes em séries temporais é extremamente prejudicial ao processo de análise de dados, pois as amostras deste tipo de base estão altamente correlacionadas e a falta de parte deles prejudica as predições feitas a partir de seu estudo, deixando o problema investigado mais complexo (BOX, JENKINS, REINSEL, 2008).

Para mitigar esta problemática, vários métodos foram propostos. A abordagem mais simples consiste na eliminação dos exemplos ou atributos que contenham valores ausentes. Todavia, este procedimento não é viável em bases com um grande número de dados faltosos, pois o descarte destas amostras gera um viés na base de dados como também uma grande perda de informação (HRUSCHKA et al., 2009b). Assim, a solução mais adequada é tratar estes valores em falta por meio de sua substituição por um valor plausível, o qual é predito a partir de um modelo computacional, processo chamado de imputação (GARCÍA-LAENCINA, SANCHO-GÓMEZ, FIGUEIRAS-VIDAL, 2010) e (GRAHAM, 2009).

Neste contexto, as heurísticas de imputação propostas são diversas e compreendem o tratamento de valores ausentes em bases de dados de diferentes naturezas. Um dos métodos mais usuais neste processo é o *k Nearest Neighbour Imputation* (KNNI). Neste algoritmo, para cada valor ausente faz-se uma busca pelas k instâncias mais similares à instância com dados faltosos, e as lacunas são preenchidas a partir da média ou moda dos valores destas k instâncias (PAN et al., 2015). Desta forma, este método pode ser usado tanto em bases com variáveis categóricas quanto em bases com valores contínuos, obtendo uma boa acurácia, contudo, ele é computacionalmente custoso, pois para cada valor ausente é necessário fazer uma busca global na base de dados (BATISTA, MONARD et al., 2002).

Outro algoritmo muito usual é o *Expectation Maximization* (EM). Inicialmente proposto por Dempster, Laird e Rubin (1977), este método calcula iterativamente o vetor média e a matriz de covariância da distribuição dos dados, com o objetivo de maximizar a função de verossimilhança da *Função Densidade de Probabilidade* da informação. Esta técnica é bastante

eficiente e produz bons resultados, contudo é um método que exige pelo menos uma variável auxiliar para estimar os valores condicionais esperados, sendo portanto uma alternativa inviável para bases de dados univariadas (GARCIA, KALENATIC, BELLO, 2010).

Além destas técnicas, algoritmos evolutivos também são muito explorados para realizar a imputação de dados devido sua robustez, por serem métodos de busca adaptativos e também por possuírem a capacidade de explorar rapidamente o espaço de busca (HRUSCHKA et al., 2009a) e (LINGRAS, ZHONG, SHARMA, 2008). A partir desta ideia, García, Kalenatic e Bello (2011) utilizam um Algoritmo Genético (AG), baseado na minimização da função de erro originada da matriz de covariância e vetores de média, para tratar valores ausentes em bases de dados multivariadas. Também neste âmbito, Lobato et al. (2015) desenvolveram um AG elitista e multi-objetivo para construir um modelo de imputação em bases de dados com atributos categóricos e numéricos simultaneamente. Tran, Zhang e Andreae (2015), por outro lado, propõe um método de imputação múltipla por meio de PG, que constrói funções de regressão para estimar valores ausentes em bases de dados multivariadas.

Aliando algoritmos evolucionários a outras abordagens computacionais, Leke, Marwala e Paul (2015) utilizam redes neurais de aprendizado profundo combinadas a algoritmos genéticos, inteligência de enxames e estimadores de verosimilhança para calcular de forma precisa valores ausentes em uma base de dados. Gautam e Ravi (2015) também usam a inteligência de enxames para atacar o problema dos dados faltosos. Os autores desenvolveram um algoritmo híbrido que combina otimização por enxame de partículas (PSO), clusterização e aprendizado de máquina auto-associativo, para construir um modelo de imputação que preserve a estrutura de covariância dos dados. Neste mesmo contexto, Aydilek e Arslan (2013) propõe um método que usa um algoritmo *fuzzy c-means* de clusterização, que combina SVR (*Support Vector Regression*) a um AG para estimar valores ausentes.

Também explorando a computação evolucionária e agora no âmbito das séries temporais, Garcia, Kalenatic e Bello (2010) propõe um algoritmo de imputação capaz de tratar valores ausentes em séries temporais univariadas. Neste trabalho, os autores codificaram um AG cuja função objetivo é baseada em três critérios estatísticos característicos de uma série temporal: a média, a variância e a auto-correlação dos dados; permitindo assim que as peculiaridades originais da série sejam preservadas, e mostrando então que AE são uma boa escolha para lidar com dados ausentes neste contexto.

Por outro lado, em relação a séries temporais multivariadas, Junger e Leon (2015) fa-

zem uso do algoritmo EM, assumindo que a base de dados tem a estrutura de uma distribuição normal, onde algoritmo é responsável por estimar o vetor média e da matriz de covariância da distribuição. Nesta implementação, além de considerar possíveis correlações entre as séries, o algoritmo considera características temporais das mesmas, tais como variações sutis entre amostras e possíveis sazonalidades. Neste trabalho, os autores obtiveram bons resultados independentemente de como as lacunas estão distribuídas, principalmente em casos onde as bases possuem até 5% de valores ausentes, se mostrando portanto um método robusto e promissor.

Outro trabalho que explora as particularidades de uma série temporal é o de Cai et al. (2015), o qual lida com três pontos característicos desta estrutura: alta ordem, que corresponde a séries temporais coletadas de diferentes fontes e que representam o mesmo processo; restrições contextuais, que diz respeito a limitações que um fenômeno possui sob determinadas condições; e variações sutis da série temporal, a qual se refere a correlação entre as amostras vizinhas. Neste trabalho, a imputação é tratada como um problema de otimização baseado em fatoração de tensão e sistemas dinâmicos multilineares. Esta abordagem oferece bons resultados tanto em termos de imputação quanto em termos de predição, sendo também um método escalável. Ademais, este artigo mostra que a análise de séries temporais pode ser vista como um problema de otimização e que portanto é passível de aplicar PG.

Apesar de bastante eficazes, a maioria dos métodos mencionados tem uma desvantagem em comum em relação aos métodos propostos neste trabalho, que é o fato de não produzirem um modelo compreensível para o tratamento dos valores ausentes. A única exceção é o algoritmo desenvolvido por Tran, Zhang e Andreae (2015). Esta é uma característica muito comum na literatura, que geralmente prioriza a acurácia da predição, contudo, algoritmos que geram modelos compreensíveis, como a programação genética, podem também obter boa acurácia para regredir os valores em falta, sendo portanto um abordagem interessante de ser explorada.

No âmbito das séries temporais, diferentemente de métodos de imputação tradicionais, o modelo de imputação dos dados deve considerar algumas das particularidades da distribuição dos dados, como a correlação entre amostras e suaves variações entre dados adjacentes. Os autores desta área, como os mencionados anteriormente, destacam a importância do uso destas características para a imputação de dados, de forma a manter as propriedades originais da distribuição. Este trabalho também faz uso destas peculiaridades para a construção dos modelos regressivos, seguindo a mesma estratégia de Garcia, Kalenatic e Bello (2010).

3. Fundamentação Teórica

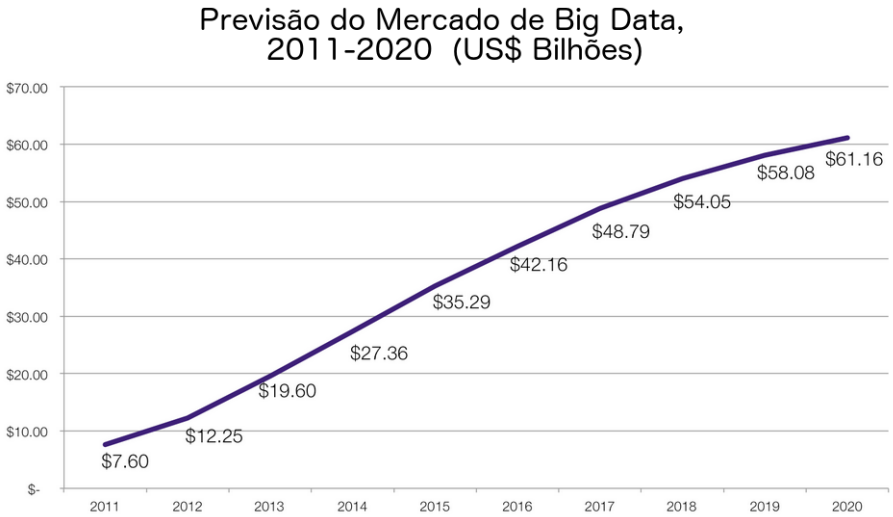
Neste capítulo são abordados os tópicos estudados que ofereceram embasamento teórico para o desenvolvimento dos algoritmos GPImpute e LGPImpute. Estes conceitos envolvem o processo de extração de conhecimento e pré-procesamento de dados, onde está incluso o principal objetivo do trabalho, que é o de estimar valores ausentes em uma base de dados de série temporal; a definição, caracterização e métodos de análise de séries temporais, que são o tipo de distribuição de dados estudada; características e funcionamento da programação genética, que é o método evolucionário utilizado para a criação de funções de regressão que estimam as amostras em falta; e por fim, uma breve introdução ao método de interpolação de Lagrange, que é a técnica utilizada no processo de pré-imputação do algoritmo LGPImpute.

3.1 O processo de extração de conhecimento (KDD)

Em virtude do crescente aumento do fluxo de dados e do barateamento das mídias de armazenamento a quantidade de informação disponível é cada vez maior. Atualmente, o conhecimento é uma ferramenta vital para se manter competitivo no mercado, de forma que dados das mais diversas naturezas se tornaram produtos comercializáveis. A Figura 3.1 retrata bem esse cenário. Em 2011 por exemplo, o mercado de Big Data correspondia a $7.6 bilhões de dólares americanos e estima-se que em 2020 este mercado alcance $61 bilhões de dólares, o que representa um crescimento de 26% ao ano (KELLY, 2014).

Apesar da vasta disponibilidade de dados e da grande necessidade em obter conhecimento a partir da análise dos mesmos, este processo não é trivial. São necessários alguns procedimentos e recursos computacionais para alcançar pleno entendimento desta informação, processo conhecido como Extração de Conhecimento ou KDD, *Knowledge-Discovery in Databases*. A Figura 3.2 ilustra o processo de KDD. Este processo é composto fundamentalmente por cinco etapas (FAYYAD, PIATETSKY-SHAPIRO, SMYTH, 1996), (HAN, KAMBER, PEI, 2011):

Figura 3.1: Previsão de Mercado de Big Data 2011-2020.



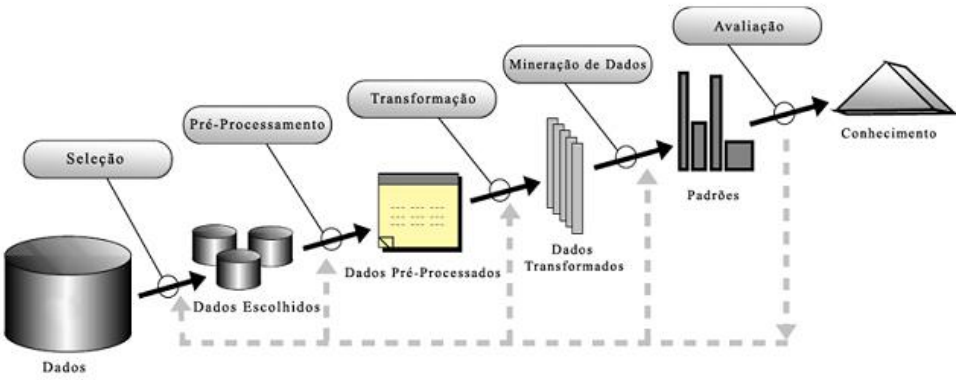
Fonte: Adaptada de Kelly (2014).

- **Seleção**: selecionar as informações relevantes a aplicação.
- **Pre-processamento**: remover ruídos e dados inconsistentes e combinar dados provenientes de várias fontes.
- **Transformação**: organizar os dados de forma apropriada para efetuar a mineração deles.
- **Mineração de Dados**: escolher o algoritmo apropriado para analisar a base de dados e extrair os padrões.
- **Avaliação**: Interpretar os padrões adquiridos através da seleção dos padrões mais relevantes fazendo com que estes sejam compreendidos por humanos.

3.2 Pré-processamento de dados

O presente trabalho encontra-se na etapa de pré-processamento, responsável pela remoção de ruídos e de dados inconsistentes. É nesta etapa que é feito tratamento de valores ausentes. Este processo é fundamental uma vez que a presença de lacunas na base de dados torna mais complexa a análise dos mesmos, de forma que o tratamento inadequado dos dados ausentes pode introduzir um viés na base de dados e conduzir os analistas a conclusões inconsistentes (WONG, CHIU, 1987). Esta seção detalha os problemas inerentes a ausência de dados e relata alguns dos procedimentos utilizados na literatura pra contornar este problema.

Figura 3.2: Processo de Extração de Conhecimento (KDD).



Fonte: Adaptada de Fayyad, Piatetsky-Shapiro e Smyth (1996).

3.2.1 Representação de uma base de dados

As colunas de uma base de dados são denominadas atributos, as linhas são chamadas de instâncias e os valores que armazenam podem ser contínuos ou discretos. De forma geral quatro tipos de atributos podem ser estudados: numéricos, os quais consistem em valores inteiros e reais; nominais, que consistem em um conjunto finito de rótulos; strings, atributos usualmente usados em mineração de texto onde as amostras podem assumir qualquer valor; data, onde as amostras assumem valores referentes a tempo no formato "yyyy-MM-dd'T'HH:mm:ss".

Após a seleção das informações relevantes a uma dada aplicação, os dados são reunidos em uma estrutura chamada base de dados para que então sejam pre-processados. Uma base de dados consiste em uma matriz $n \times d$ que contém informações a respeito de um fenômeno estudado. Nesta matriz as linhas armazenam as informações inerentes a um exemplo/amostra e cada coluna representa um aspecto observado (ZAKI, JR, 2014). Quando esta matriz é $n \times 1$ a base de dados é considerada *univariada* pois contém apenas um aspecto que caracteriza o fenômeno analisado. Bases de dados $n \times 2$, por outro lado, são chamadas *bivariadas*, estas estruturas contém dois atributos que representam o fenômeno estudado e estes possivelmente estão correlacionados. Por fim, matrizes com mais de 3 colunas consistem de bases de dados *multivariadas*, vários atributos caracterizam o fenômeno observado e estes podem possuir alguma relação ou não. A Figura 3.3 ilustra exemplos bases univariada, bivariada e multivariada. Nestes exemplos, as amostras assumem valores categóricos e numéricos onde os valores em falta são representados pelo carácter "?".

Figura 3.3: Exemplos de base de dados.

A1
10
?
-3.7
?
?
67
5.4

(a) base de dados univariada

A1	A2
1	b
5	a
7	?
?	b
8	b
?	a
14	a

(b) base de dados bivariada

A1	A2	A3	A4
1	sim	10	3.1
2	nao	?	5.6
3	?	76	7.1
4	nao	101	9.6
?	?	65	3.4
6	sim	?	5.7
7	nao	11	9.7

(c) base de dados multivariada

Fonte: Elaborada pela autora.

3.2.2 O problema dos valores ausentes

Valores ausentes são pontos desconhecidos a respeito do fenômeno estudado que provocam lacunas na base de dados. Esta ausência de informação é devido a vários fatores, dentro os quais destacam-se: o mal funcionamento de sensores que como resultado deixam de adquirir os dados desejados, a troca do modelo de aquisição de dados que pode gerar perdas de informação, ou mesmo a falta de respostas em questionários de pesquisas (GRAHAM, 2009).

Segundo Barnard e Meng (1999), três problemas estão comumente associados à presença de valores ausentes, são estes: (i) perda de eficiência; (ii) problemas para manipular e analisar os dados; (iii) viés proveniente das diferenças entre os dados completos e os dados faltantes. Em aplicações de classificação, a falta de dados pode inclusive inviabilizar o processo, pois a maioria dos algoritmos deste domínio não conseguem lidar diretamente com dados ausentes, além de que lacunas na base de dados afetam a acurácia de predição do classificador (GHEYAS, SMITH, 2010).

Em análise de séries temporais, dados faltosos também afetam consideravelmente a pesquisa, tornando-a mais complexa, pois devido as amostras estarem altamente correlacionadas, a presença de lacunas na série quebra características estatísticas importantes referentes à distribuição dos dados, afetando sua modelagem (BOX, JENKINS, REINSEL, 2008).

3.2.3 Tratamento de valores ausentes

A forma mais simples, porém menos eficiente de lidar com lacunas na base de dados, é ignorar os atributos e instâncias que apresentem valores em falta. No entanto, esta abordagem é válida apenas para bases de dados com poucos valores ausentes, de forma que quanto maior a quantidade de dados faltosos, maior a perda de informação, como também o viés introduzido

nas análises estatísticas (HRUSCHKA et al., 2009b). Visando mitigar este problema, diversos métodos foram desenvolvidos para estimar dados faltosos, processo chamado de substituição de valores ou imputação.

De acordo com Luengo, García e Herrera (2012), é importante analisar a distribuição dos dados ausentes e o padrão que estes seguem para escolher de forma adequada um mecanismo de imputação, caso esta seja possível. Little e Rubin (1987) apontam que há três mecanismos de indução de valores ausentes, são estes:

1. **MCAR**: do inglês *Missing Completely at Random*, ou ausência completamente aleatória, ocorre quando a probabilidade de haver um dado ausente na base não depende nem dos valores conhecidos nem dos valores faltantes. Nesse tipo de distribuição, qualquer método de imputação pode ser aplicado sem que haja riscos de introduzir um viés na base.

2. **MAR**: do inglês *Missing at Random*, ou ausência aleatória, ocorre quando a probabilidade de ocorrer um valor ausente depende de um valor conhecido mas não do dado faltante em si. Nesse caso, os dados ausentes não estão distribuídos de forma aleatória em toda a base de dados mas estão aleatoriamente distribuídos em um ou mais atributos.

3. **NMAR**: do inglês *Not Missing at Random*, ou ausência não aleatória, ocorre quando a probabilidade de um valor ausente pode depender do valor do atributo. É o tipo de distribuição mais complexa de lidar e os dados em falta não devem ser ignorados.

O algoritmo de imputação a ser aplicado depende da natureza dos dados. Em atributos categóricos são usados algoritmos de árvore de decisão, estatísticos e de agrupamento. Já no caso de atributos numéricos, além de métodos estatísticos e de agrupamento, são comumente usados algoritmos de regressão. O software KEEL (LUKE et al., 2004) disponibiliza diversos algoritmos de imputação de dados. Este trabalho lida com três deles de forma a comparar o seu desempenho com o método proposto, são eles: KMI, MC e EC. Estes algoritmos foram escolhidos pois representam três esferas de métodos de imputação: substituição baseada em agrupamento, substituição estatística e substituição pelo vizinho mais próximo. Os algoritmos estudados são brevemente descritos a seguir.

- **EC**: agrupa as amostras em *clusters* baseado em sua interdependência e extrai os padrões inerentes a cada *cluster* para estimar os valores ausentes (WONG, CHIU, 1987).

- **MC**: no caso de atributos numéricos, valores ausentes são simplesmente substituídos pela média da distribuição no atributo tratado. No caso de atributos categóricos, estes valores são substituídos pela moda da distribuição (BATISTA, MONARD, 2003).

- **KMI**: divide a base em *clusters* baseado na similaridade dos dados, de forma a minimizar a dissimilaridade inerente a cada *cluster*, a qual é calculada baseada na distância entre seu centro e cada amostra pertencente a ele. O centro do *cluster* é dado pela média de seus valores; a distância é dada pela distância euclidiana entre um valor e outro. O dado ausente é substituído pelo vizinho que possuir a menor distância (LI et al., 2004).

3.3 Séries Temporais

Uma série temporal é um sequência de observações feitas ao longo do tempo de forma sucessiva. São pontos discretos provenientes do estudo de um dado fenômeno ou processo que estão, na maioria dos casos, igualmente espaçados no domínio do tempo e que normalmente possuem alguma dependência com amostras adjacentes (BOX, JENKINS, REINSEL, 2008).

Este tipo de estrutura está presente em diversos domínios de aplicação a exemplo das ciências naturais como a meteorologia, que analisa diariamente dados do ambiente a fim de fazer predições do clima em uma região; ciências biológicas como a medicina, que faz observações periódicas de sinais biológicos e fisiológicos de um paciente com o objetivo de estudar o avanço de uma doença; engenharia, que estuda o comportamento de processos a fim de identificar falhas e maneiras de otimiza-lo; e economia, que analisa dados financeiros de empresas e países para verificar riscos de investimento.

Dada a grande variedade de aplicações de séries temporais, Box, Jenkins e Reinsel (2008) as dividem em cinco domínios importantes:

1. **Predições**: busca determinar o valor de uma observação no tempo t a partir das observações conhecidas no tempo t - z.

2. **Determinação de funções de transição**: faz um estudo das amostras de entrada e saída de um processo a fim de entender os efeitos ocorridos ao longo do mesmo.

3. **Análise de intervenções**: verifica as consequências de eventos externos em um processo.

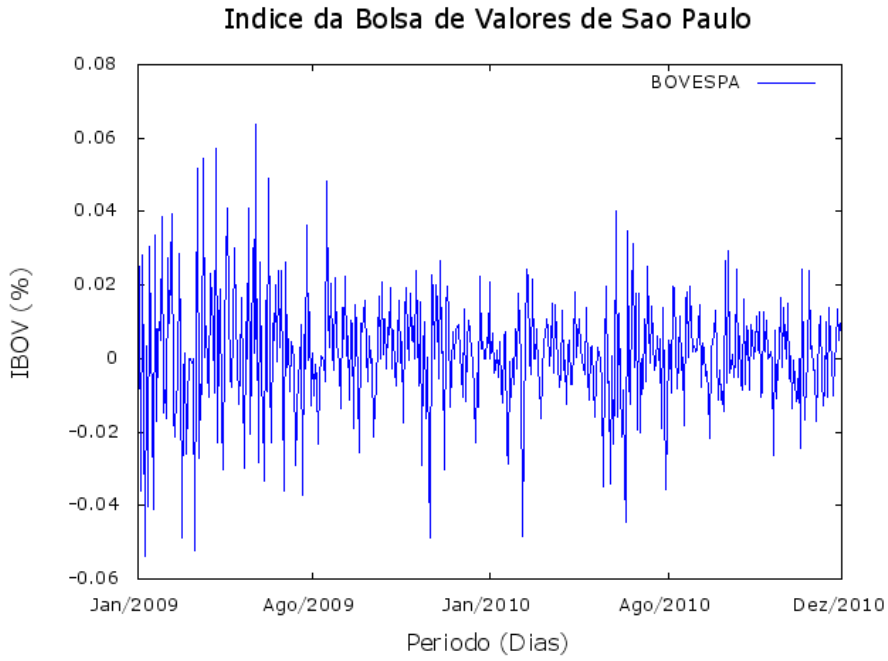
4. **Análise de inter-relações**: estuda correlações entre séries temporais que estejam relacionadas (séries temporais multivariadas).

5. **Projeto de sistemas de controle**: analisa mudanças em um processo visando compreender o porquê delas ocorrerem.

Dependendo da aplicação estudada, o número de fatores que influencia o fenômeno observado varia. Em casos onde apenas uma característica é responsável por definir o processo, uma única série temporal é formada. Estas são ditas séries temporais *univariadas*. Já no caso de processos definidos por vários fatores simultaneamente, haverá mais de uma série temporal representando os dados observados. Estas são ditas séries temporais *multivariadas*, onde cada série consiste de informações relacionadas a um aspecto do fenômeno estudado e o conjunto delas representa todo o processo, estando as mesmas correlacionadas de alguma maneira.

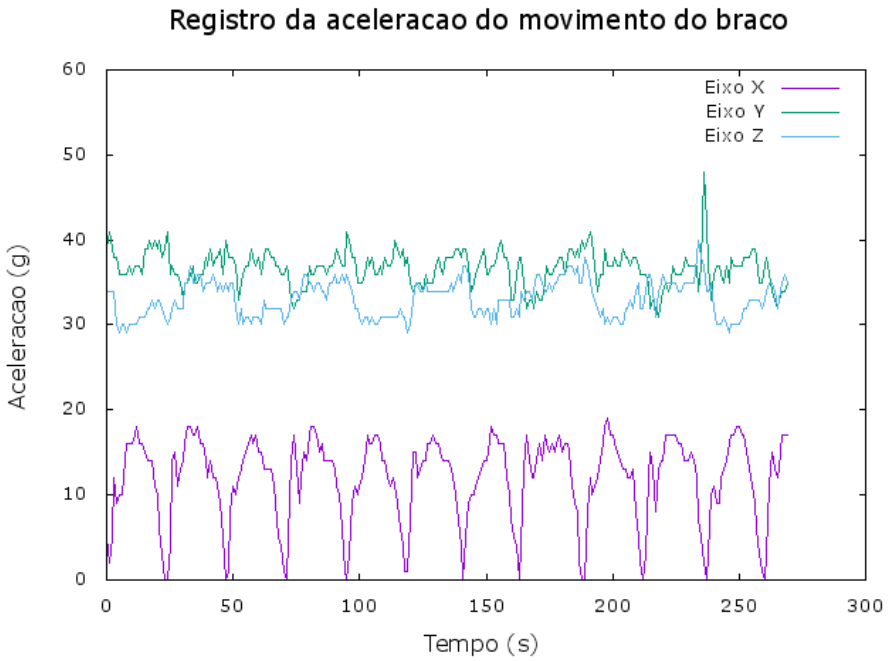
As Figuras 3.4 e 3.5 ilustram séries temporais univariada e multivariada respectivamente. Ambas foram extraídas do repositório UCI (LICHMAN, 2013) de aprendizado de máquina, a primeira da base de dados *Istanbul Stock* e a segunda da base de dados *ADL Climb Stairs*. A Figura 3.4 representa os dados da bolsa de valores de São Paulo entre Janeiro de 2009 e Dezembro de 2010, onde cada observação corresponde ao índice da BOVESPA em um dia útil. Já no caso da Figura 3.5, há três séries temporais contendo informações de um único processo: o movimento do braço ao subir uma escada. Cada série temporal é referente a um dos eixos analisados por um acelerômetro tri-axial atrelado ao pulso de uma mulher.

Figura 3.4: Índice da bolsa de valores de São Paulo.



Fonte: Elaborada pela autora.

Figura 3.5: Movimentação do braço ao subir uma escada.



Fonte: Elaborada pela autora.

Tendo em mente a gama de aplicações de séries temporais, o presente trabalho foca em dois domínios: predições e análise de interrelações; de tal forma que os algoritmos codificados exploram séries temporais multivariadas, com o objetivo de utilizar as relações entre as mesmas para gerar um modelo de predição capaz de estimar os valores ausentes na série.

3.3.1 Análise de Série Temporal

Segundo Shumway e Stoffer (2010), a correlação existente entre pontos adjacentes em uma série temporal pode restringir a aplicabilidade de métodos tradicionais de análise de dados, pois estes métodos geralmente assumem que as amostras de dados são independentes entre si e estão uniformemente distribuídas.

Tendo isto em mente, a análise de série temporal deve considerar alguns aspectos inerentes a esse tipo de estrutura, tais como a correlação entre amostras adjacentes, possíveis sazonalidades e a suavidade da variação entre amostras (BOX, JENKINS, REINSEL, 2008), (SHUMWAY, STOFFER, 2010), (GARCIA, KALENATIC, BELLO, 2010) e (CAI et al., 2015). Este estudo é tipicamente feito no domínio do tempo ou no domínio da frequência, e vários modelos estocásticos foram criados para representar as diferentes maneiras que os dados estão distribuídos.

Em aplicações de predição e controle de séries temporais, dois modelos estocásticos são

considerados, o modelo estacionário e o modelo não estacionário. No caso de processos estacionários, as propriedades estatísticas da série temporal não variam ao longo do tempo e portanto valores como média e variância se mantem constantes. Por outro lado, os processos não estacionários são desequilibrados estatisticamente, de forma que estes valores variam com o tempo (SHUMWAY, STOFFER, 2010).

AR, MA, ARMA e ARIMA são exemplos de modelos estocásticos que podem representar uma série temporal. O primeiro é um modelo de auto-regressão onde a combinação linear de valores passados na série, somada a uma perturbação qualquer determinam o valor presente. O valor atual do modelo MA (*Moving Average*) em contrapartida, é representado por uma soma finita e ponderada das perturbações impostas nas observações anteriores.

O modelo ARMA (*Mixed Autoregressive-Moving Average Models*), por outro lado, é a junção dos modelos AR e MA. Esta mesclagem garante maior flexibilidade à representação e abrange um maior número de séries temporais, contudo é restrita a modelos estocásticos estacionários. O modelo ARIMA (*Autoregressive Integrated Moving Average*) no entanto, pode representar tanto modelos estacionários quanto modelos não estacionários, e é um dos mais usados para estudar séries temporais, pois abrange uma grande gama delas. O ARIMA, além de considerar as questões investigadas pelos outros modelos, também leva em consideração flutuações que ocorrem ao longo da série em tempos diferentes e que podem se repetir de alguma forma (BOX, JENKINS, REINSEL, 2008). Devido sua vasta aplicabilidade e generalidade, as principais medidas estatísticas utilizadas neste modelo foram escolhidas para caracterizar as séries temporais estudas neste trabalho, de forma que este modelo pode ser aplicado após a imputação dos dados para extrair informação.

3.3.2 Estatísticas Relevantes

Como dito anteriormente, séries temporais possuem especifidades que devem ser consideradas ao trabalhar com este tipo de estrutura, a citar: (i) sazonalidade; (ii) correlação entre estruturas adjacentes, (iii) variações sutis entre observações e (iv) correlação com outras séries temporais (BOX, JENKINS, REINSEL, 2008), (SHUMWAY, STOFFER, 2010), (GARCÍA, KALENATIC, BELLO, 2010) e (CAI et al., 2015).

Para lidar com estes aspectos, o presente trabalho faz uso de um algoritmo de Programação Genética (PG) que tem como objetivo gerar uma função de regressão para estimar valores ausentes, de forma que as características originais da série não sejam alteradas.

Para manter a integridade da estrutura, três dos parâmetros principais usados para construir o modelo de representação ARIMA foram considerados: média, variância e autocorrelação entre as amostras. Esta metodologia é baseada no trabalho de Garcia, Kalenatic e Bello (2010), que utiliza estes parâmetros para fazer o tratamento de valores ausentes em séries temporais univariadas usando um AG.

Consideremos então um processo estocástico constituído de uma sequência de n observações, dada por x, tal que $x \in S$ e S é o espaço que contém os valores que x pode assumir. Cada uma destas observações é feita em um tempo t, tal que $t \in T$ e T é espaço de tempo descrito pela sua função densidade de probabilidade.

Desta forma, a média de x, definida na Equação 3.1, é calculada a partir da soma de todas as amostras da série dividida por n.

$$\mu_x = \frac{1}{n}\sum_{t=1}^{n} x_t \tag{3.1}$$

A variância, em contrapartida, é calculada a partir do módulo da soma de cada observação subtraída da média da série temporal. Este cálculo é representado pela Equação 3.2.

$$\vartheta_x = \frac{1}{n-1}\sum_{t=1}^{n}(x_t - \mu_x)^2 \tag{3.2}$$

Já a autocorrelação de um processo estocástico é representada pela distância de uma observação x_t, feita em um tempo t, para uma observação x_{t+h}, feita h unidades de atraso em relação a t. Esta autocorrelação é medida a partir da razão entre a auto-covariância de x_t com uma observação atrasada por h, pela auto-covariância de x_t com ela mesma, ou seja, sem atraso. A auto-covariância e autocorrelação são dadas pelas Equações 3.3 e 3.4 respectivamente.

$$\gamma_(h) = \frac{1}{n}\sum_{t=1}^{n-|h|}(x_{t+|h|} - \mu_x)(x_t - \mu_x), \quad -n < h < n. \tag{3.3}$$

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)}, \quad -n < h < n. \tag{3.4}$$

3.4 Programação Genética

A Programação Genética (PG) foi desenvolvida na década de 90 nos Estados Unidos da América por John Koza e apresentada em Koza (1992). Ela um método de computação

evolucionária usado para otimização de problemas complexos sendo tipicamente aplicado em tarefas de aprendizado de máquina como predição e classificação, operações fundamentais na área de mineração de dados (EIBEN, SMITH, 2003). Essa técnica é utilizada para resolver diversos problemas práticos, obtendo resultados extremamente satisfatórios. A PG inclusive deu origem a novas invenções patenteadas (POLI et al., 2008).

Este tipo de abordagem é capaz de resolver problemas automaticamente, sem requerer que o usuário saiba ou especifique a forma da estrutura da solução de antemão. Em seu nível mais abstrato, é um método sistemático independente de domínio que possibilita computadores resolver problemas automaticamente a partir de uma premissa de alto-nível (POLI et al., 2008). Segundo Tran, Zhang e Andreae (2015), por meio da PG é possível aprender definições de funções baseado em exemplos de dados, o que faz dessa metodologia uma excelente alternativa como método de regressão. Ademais, este algoritmo tem a vantagem de codificar os indivíduos em uma estrutura não linear, podendo esta ser uma árvore ou um grafo, sendo portanto o método evolucionário mais viável para lidar com problemas desta natureza. O Algoritmo 1 mostra os passos básicos seguidos por PG.

Assim como em outros algoritmos evolucionários, a PG é baseada na premissa de Darwin "O mais apto sobrevive". Este método transforma estocasticamente populações de programas geração a geração, buscando criar indivíduos melhores. Vale salientar que pelo fato desta técnica ser essencialmente aleatória, ela não garante o melhor resultado, todavia esta característica randômica permite que o algoritmo contorne obstáculos que métodos determinísticos possam não conseguir desviar (POLI et al., 2008).

Algoritmo 1: Programação Genética Canônica.

```
INÍCIO
   INICIALIZAR população com soluções aleatórias;
   AVALIAR cada candidato;
   ENQUANTO (CRITÉRIO DE PARADA não for satisfeito) FAÇA
      EXECUTAR cada candidato;
      SELECIONAR pais;
      APLICAR OPERAÇÕES GENÉTICAS para criar novos indivíduos;
   FIM FAÇA
FIM
```

Fonte: Eiben e Smith (2003).

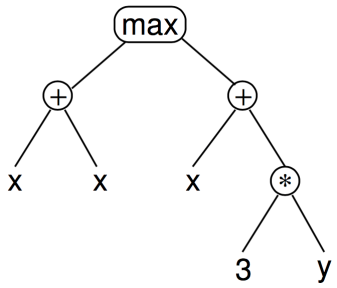
3.4.1 A estrutura do PG

Como dito anteriormente, na PG os indivíduos são tipicamente estruturas não lineares representadas por árvores ou grafos. Esta característica o faz ser um método evolucionário ímpar e portanto possui algumas diferenças em relação a manipulação dos programas (indivíduos) e dos operadores genéticos. Os tópicos a seguir detalham o funcionamento de uma PG representada por árvores.

3.4.1.1 Representação de indivíduos

Como outros métodos evolutivos, cada candidato em uma PG representa uma possível solução para o problema, mas neste caso cada indivíduo constitui de operações que o programa executa. A Figura 3.6 mostra um programa que representa a função $max(x + x, x + 3 * y)$. Observa-se que os nós internos da árvore são compostos por operações e que as folhas consistem de variáveis. Em PG estes nós são chamados de *funções* e as folhas de *terminais*.

Figura 3.6: Exemplo de indivíduo em PG.



Fonte: Poli et al. (2008).

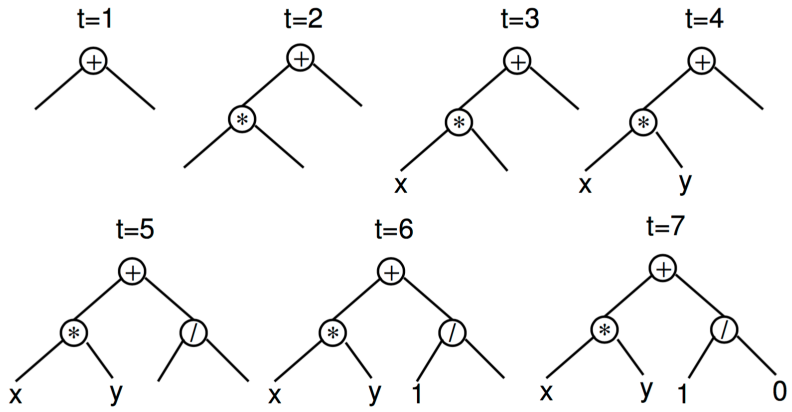
3.4.1.2 Inicialização da população

A população de uma GP é inicializada aleatoriamente e requer que haja um grande número de indivíduos para que seja eficiente, sendo esta na ordem de milhares (EIBEN, SMITH, 2003). Há inúmeras formas de inicializar a população, Poli et al. (2008) relata as mais usadas: completa (*full*), crescida (*grow*) e meio-a-meio (*Ramped half-and-half*).

Tanto na inicialização pelo método completo, quanto pelo método de crescimento, nós são adicionados de forma aleatória até que um determinado nível de profundidade seja atingido, onde este deve ser menor ou igual ao nível máximo da árvore. No primeiro caso, ilustrado na Figura 3.7, as árvores são geradas de forma que todas as folhas da árvore possuam o mesmo

nível de profundidade, e todos os nós pais possuam o mesmo número de filhos. Neste processo, são adicionados aleatoriamente novos nós (funções), seguido de todos os seus filhos, que no último nível da árvore são chamados folhas (terminais).

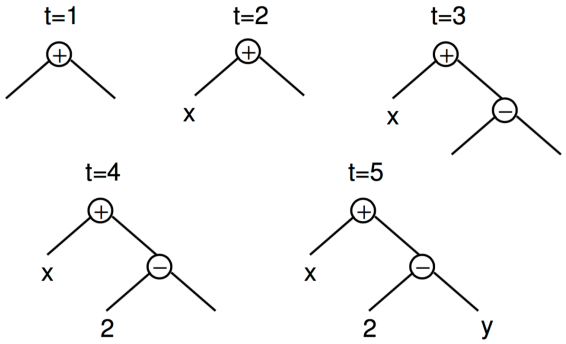
Figura 3.7: Processo de geração de indivíduo pelo método completo com profundidade igual a 2.



Fonte: Poli et al. (2008).

Por outro lado, no método de crescimento, mostrado na Figura 3.8, apesar dos elementos também serem adicionados aleatoriamente e até que atinjam um certo nível de profundidade, uma vez que este seja atingido, apenas terminais podem ser adicionados a estrutura, ou seja, as folhas da árvore não precisam estar no mesmo nível. Neste processo, uma função é adicionada seguida da outra, sem a preocupação que cada nó possua todos os filhos, e uma vez que o nível máximo da árvore seja atingido, apenas terminais são adicionados a estrutura.

Figura 3.8: Processo de geração de indivíduo pelo método de crescimento com profundidade igual a 2.



Fonte: Poli et al. (2008).

O método meio-a-meio, em contrapartida, é uma técnica usada para garantir maior variabilidade genética a população. Nesta abordagem, metade dos indivíduos são gerados pelo

método completo e a outra metade pelo método de crescimento.

3.4.1.3 Operadores Genéticos

Os operadores genéticos de uma GP são basicamente três: seleção, recombinação e mutação. A seleção pode ser feita por qualquer mecanismo usado em algoritmos evolucionários tais como seleção proporcional a função de aptidão, amostragem universal estocástica ou por torneio (POLI et al., 2008) sendo que o último é mais comumente utilizado. Neste método, um número de indivíduos da população é escolhido aleatoriamente e são avaliados de acordo com a função de aptidão. Os dois indivíduos mais aptos são então selecionados como pais e são recombinados para gerar a nova população. De acordo com Eiben e Smith (2003), para aumentar a eficiência, a quantidade de indivíduos selecionados para serem pais é normalmente proporcional ao tamanho da população. Por convenção, para uma população de 1000 indivíduos a seleção é de 32%, no caso de 2000 é de 16%, 4000 é de 8% e assim por diante.

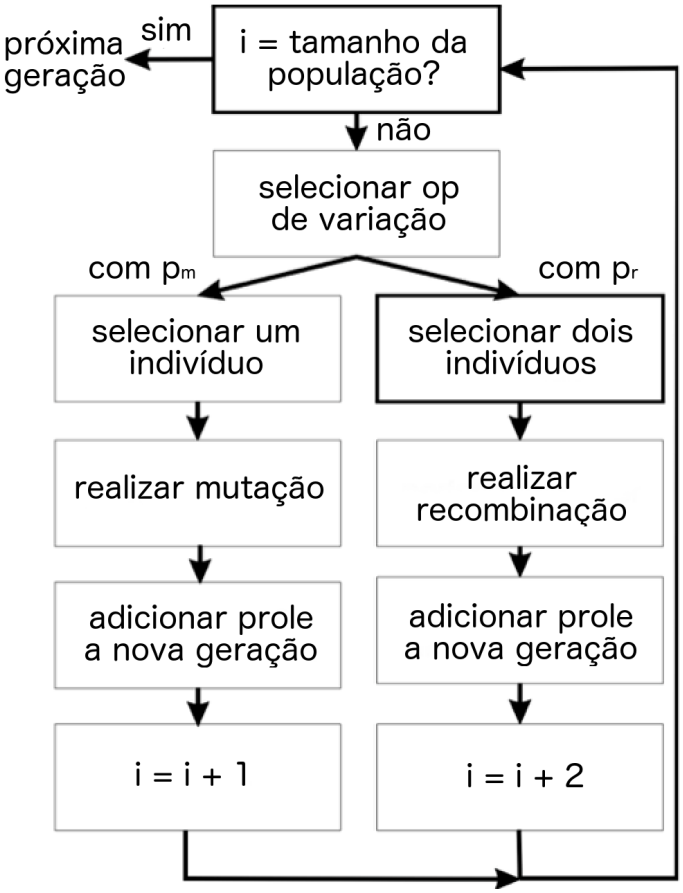
A recombinação e mutação, chamadas de operações de variabilidade genética, são definidas probabilisticamente e de forma mutualmente exclusiva, ou seja, jamais ocorrem ambas na mesma geração (EIBEN, SMITH, 2003), (POLI et al., 2008), (KOZA, 1992). Estas operações possuem duas probabilidades, a primeira corresponde à probabilidade que ela ocorra na geração, p_m no caso da mutação e p_r no caso da recombinação, e a segunda diz respeito à probabilidade de escolher um ponto no indivíduo que sofrerá a transformação.

Geralmente, a recombinação possui a maior probabilidade, sendo esta 90% ou maior. Já no caso da mutação, autores recomendam que esta não ultrapasse um certo limite. De acordo com Koza (1992) este não deve ultrapassar 1%. Banzhaf et al. (1998), em contrapartida, defende que este parâmetro não seja maior a 5%. O fluxo de reprodução de uma PG é ilustrado na Figura 3.9. Nele em cada geração da PG um operador de variabilidade é usado. Dependendo da probabilidade de ocorrência, p_m ou p_r, os indivíduos selecionados sofrerão mutação ou serão recombinados.

Caso a operação de mutação seja escolhida, um único indivíduo da população é selecionado para ser modificado. Após este processo ele é adicionado à população da nova geração. Caso a operação escolhida seja de recombinação, 2 indivíduos são selecionados, seus materiais genéticos são trocados, e ambos são adicionados a nova população. Isto ocorre até que o número total de indivíduos i da próxima geração seja alcançado.

Os operadores de variabilidade genética geram sua prole a partir da escolha de um ponto

Figura 3.9: Fluxo de reprodução de PG.

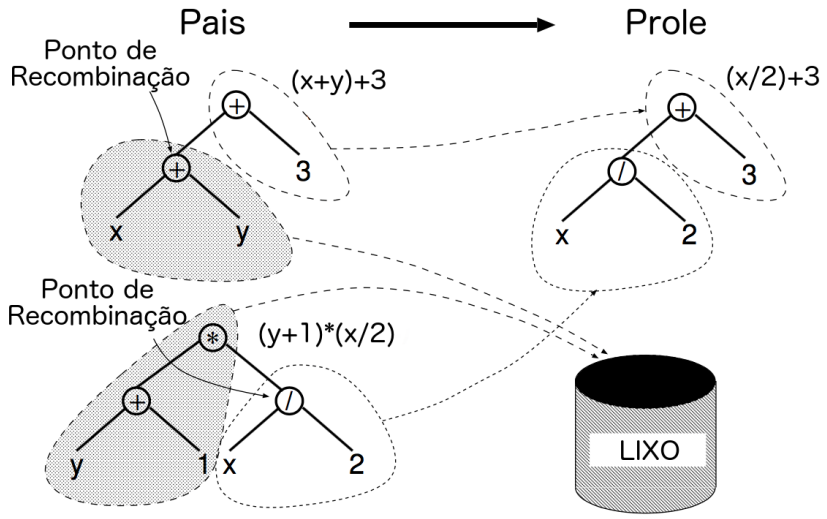


Fonte: Adaptada de Eiben e Smith (2003).

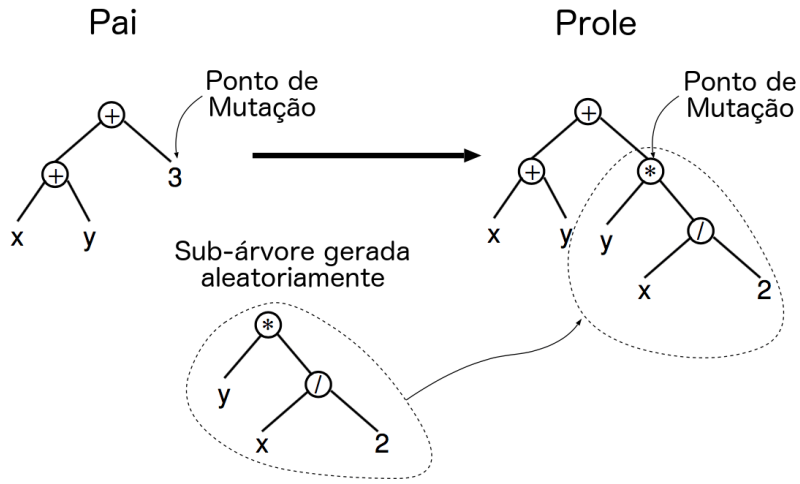
de alteração no indivíduo. Na recombinação, dois pais são selecionados e um ponto de cruzamento é definido em cada um deles, para que as sub-árvores geradas sejam então combinadas e o material genético excedente seja descartado. Na mutação, a sub-árvore resultante da poda é descartada e uma nova sub-árvore é randomicamente gerada. A Figura 3.10 mostra como estas operações são feitas.

Na recombinação, a partir de um nó específico, denominado ponto de recombinação, a árvore que representa o indivíduo é dividida em duas sub-árvores em cada programa selecionado, duas das sub-árvores combinadas gerarão o novo indivíduo e resto do material genético é jogado fora. No processo de mutação, um nó é escolhido como ponto de mutação. A partir deste ponto uma subárvore é gerada aleatoriamente e o novo indivíduo gerado é adicionado à nova geração.

Figura 3.10: Operações de recombinação e mutação em um PG.



(a) Processo de recombinação em dois indivíduos na PG



(b) Processo de mutação em um indivíduo na PG

Fonte: Adaptada de Poli et al., 2008.

3.4.1.4 Funções e Terminais

Dependendo do problema atacado, os elementos que compõe os indivíduos de uma PG mudarão. Os terminais dos programa devem ser de três tipos: variáveis de entrada externas como x e y por exemplo; funções sem argumentos como $rand()$ e $mover\_cima()$; e valores constantes. As funções, por outro lado, são primitivas que variam de operadores aritméticos como $+,-,*,/$ e operadores lógicos como AND, FOR e IF-THEN-ELSE. A Tabela 3.1 sumariza

as primitivas usadas para formar estes conjuntos.

Tabela 3.1: Exemplo de primitivas usadas em PG.

Conjunto de Funções	
Tipo de Primitiva	*Exemplo(s)*
Aritmética	+,-,*,/
Matemática	sen, cos, exp
Booleana	AND, OR, NOT
Condicional	IF-THEN-ELSE
Laços	FOR, REPEAT
⋮	⋮
Conjunto de Terminais	
Tipo de Primitiva	*Exemplo(s)*
Variáveis	x, y
Valores Constantes	3; 0,45
Funções sem parâmetros	$rand()$, $vire\_esquerda()$

Fonte: Adaptada de Poli et al., 2008.

3.5 Interpolação

Neste trabalho, a interpolação é usada para criar uma base de dados temporária que auxiliará a programação genética na geração de funções de regressão. Esse processo consiste de uma pré-imputação realizada em cada trecho da base de dados com valores ausentes. Nestas lacunas, um polinômio específico é estimado para determinar as amostras em falta na série temporal, de forma que a nova base de dados gerada oferecerá suporte no cálculo referente a função de cada indivíduo, que, por utilizar dados de atributos vizinhos a série temporal sendo regredida, pode gerar um número indefinido. Porém, a interpolação impede que isto aconteça.

De forma geral, a interpolação é utilizada para simplificar funções muito complexas aproximando-as à funções polinomiais, como também para determinar funções desconhecidas onde é sabido apenas alguns dos pontos discretos que as compõe. De acordo com o teorema de Weierstrass, "Toda função contínua pode ser arbitrariamente aproximada por um polinômio". Esta é uma abordagem simples que oferece várias vantagens, pois polinômios são facilmente computáveis, suas derivadas e integrais são também polinômios e suas raízes são relativamente fáceis de determinar (FRANCO, 2006).

Assim, a partir de $n + 1$ pontos distintos, reais ou complexos, $w_0, w_1, w_2, ..., w_{n+1}$ pertencentes ao domínio da função $f(w)$, e os pontos $y_0, y_1, y_2, ..., y_{n+1}$, reais ou complexos, que

compõe a imagem da função $f(w)$, é possível determinar um único polinômio p de grau no máximo n tal que $p(w_0) = y_0, p(w_1) = y_1, ..., p(w_n) = y_n$ (FRANCO, 2006). Este polinômio é chamado de polinômio interpolador e é definido por:

$$p(w_j) = f_j, \qquad j = 0, ..., n \tag{3.5}$$

3.5.1 Interpolação de Lagrange

Um dos métodos eficientes para encontrar o polinômio interpolador de um conjunto de pontos é por meio da Fórmula de Lagrange (LAGRANGE, 1812), determinada por:

$$p(w) = \sum_{j=0}^{n} f_j L_j(w), \tag{3.6}$$

onde L_j é chamado de Polinômio de Lagrange e é definido por:

$$L_j(w) = \frac{\prod_{k=0,k\neq j}^{n}(w - w_k)}{\prod_{k=0,k\neq j}^{n}(w_j - w_k)}, \tag{3.7}$$

de forma que no ponto w_j, $L_j(w)$ obedece a seguinte propriedade:

$$L_j(w_k) = \begin{cases} 1, & j = k, \\ 0, & \text{caso contrário}, \end{cases} \qquad j, k = 0, ..., n. \tag{3.8}$$

Quanto maior o grau do polinômio interpolador, mais custoso é computar os coeficientes do mesmo. Berrut e Trefethen (2004) destacam três problemas inerentes a Interpolação de Lagrange em sua forma básica: (i) cada avaliação de $p(w)$ requer $O(n^2)$ adições e multiplicações; (ii) adicionar um novo par ordenado (w_{n+1}, f_{f+1}) requer que $p(w)$ seja computado novamente a partir do zero; e (iii) os cálculos são numericamente instáveis.

Apesar da complexidade computacional proveniente da Fórmula de Lagrange este método é bastante simples conceitualmente. Tendo em vista que na aplicação estudada a interpolação considerará no máximo quatro pontos em cada trecho analisado, a Equação 3.6 foi especializada para $n \leq 4$, assim tem-se:

$$L_0 = \frac{(w - w_1)(w - w_2)(w - w_3)}{(w_0 - w_1)(w_0 - w_2)(w_0 - w_3)} \tag{3.9}$$

$$L_1 = \frac{(w - w_0)(w - w_2)(w - w_3)}{(w_1 - w_0)(w_1 - w_2)(w_1 - w_3)} \tag{3.10}$$

$$L_2 = \frac{(w - w_0)(w - w_1)(w - w_3)}{(w_2 - w_0)(w_2 - w_1)(w_2 - w_3)} \tag{3.11}$$

$$L_3 = \frac{(w - w_0)(w - x_1)(w - x_2)}{(w_3 - w_0)(w_3 - w_1)(w_3 - w_2)} \tag{3.12}$$

$$p(4) = f_0 L_0 + f_1 L_1 + f_2 L_2 + f_3 L_3 \tag{3.13}$$

A partir da Equação 3.13 é possível isolar os termos referentes a cada coeficiente do polinômio e dessa forma diminuir a complexidade dos cálculos para $O(n)$ operações de adição e multiplicação. Este procedimento é feito para 2, 3 e 4 pontos. Esta técnica foi escolhida devido a simplicidade inerente a sua implementação e também seu baixo custo computacional. Outra possível abordagem para reduzir o número de operações computadas é usar a Interpolação de Lagrange em sua Forma Baricêntrica, como demonstrado em (BERRUT, TREFETHEN, 2004).

4. Métodos Propostos

Os algoritmos propostos neste trabalho, chamados de GPImpute e LGPImpute, baseiam-se em duas premissas: (i) uma série temporal é um processo estocástico onde, para manter as propriedades originais da distribuição, deve-se considerar características como média, variância e auto-correlação ao tratar valores ausentes; e (ii) um algoritmo de programação genética é capaz de regredir uma função de maneira efetiva baseado em dados de exemplo. A ideia por trás desta abordagem é descobrir, a partir de um conjunto de funções aleatórias, qual a equação que melhor define a série temporal estudada. Esta busca é feita por PG que, por ser um algoritmo evolucionário, é bastante flexível e varre o espaço de busca de forma relativamente rápida.

Ambos métodos foram desenvolvidos a partir do *framework* ECJ (LUKE et al., 2004), uma ferramenta codificada em Java que disponibiliza a implementação de vários algoritmos evolucionários. É importante frisar que no caso da programação genética, o algoritmo gera como resultado um modelo compreensível que pode ser usado para análise de padrões. Como consequência, tanto o GPImpute quanto o LGPImpute possuem esta característica. Utilizou-se também a API Weka (HALL et al., 2009) para a leitura e manipulação das bases de dados, todas no formato *arff*.

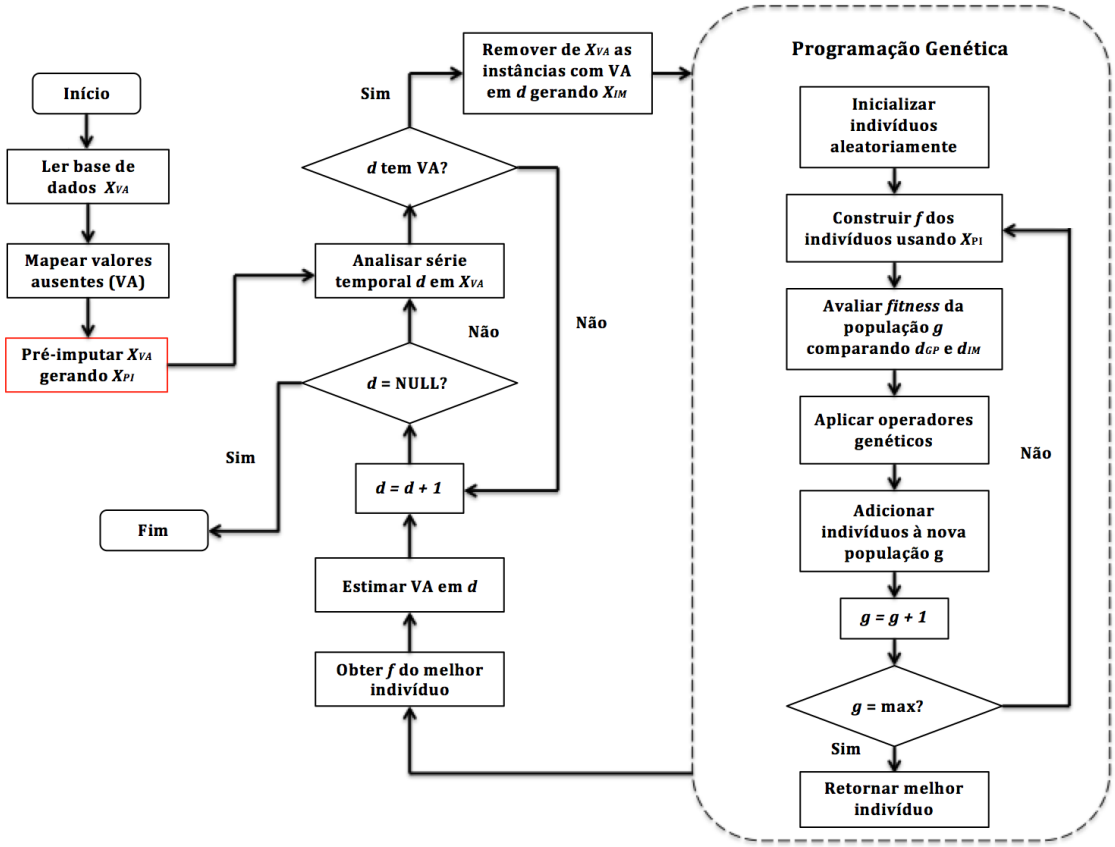
A diferença entre o GPImpute e o LGPImpute está na forma que estes algoritmos geram a base de testes usada na programação genética. O primeiro realiza a pré-imputação tendo como base as amostras vizinhas à cada lacuna, e o segundo o faz baseado em polinômios interpoladores de Lagrange, por isso o "L"no nome. Os detalhes destes métodos são descritos nas próximas seções.

4.1 O algoritmo de imputação

Como dito anteriormente, a imputação de dados é construída sobre um algoritmo de programação genética que utiliza uma função de aptidão multi-critério para avaliar os in-

divíduos. O diagrama ilustrado na Figura 4.11 mostra o funcionamento geral dos algoritmos desenvolvidos. Ambos funcionam da mesma maneira no que diz respeito a manipulação dos dados e regressão das séries temporais. O que diferencia o GPImpute e o LGPImpute é a forma com a qual eles fazem a análise dos dados, etapa destacada em vermelho no diagrama. Esta etapa corresponde a pré imputação dos dados e é responsável por fornecer dados válidos para construir as funções que cada individuo representa na PG.

Figura 4.11: Diagrama geral do fluxo dos algoritmos GPImpute e LPGImpute.



Fonte: Elaborada pela autora.

O processo de imputação começa com a leitura de uma base de dados com valores ausentes X_{VA} e o mapeamento destes valores. O próximo passo consiste na pré imputação dos dados, que tem como função construir uma base completa X_{PI} que possa ser utilizada para calcular o resultado das funções indicadas por cada indivíduo na PG. Isto é necessário porque os indivíduos da PG utilizam, em seus terminais, amostras de outros atributos na base dados. Por exemplo, se uma série temporal d está sendo regredida a partir da função $x[d-1] \times x[d+2] + 3$, caso as amostras $x[d-1]$ ou $x[d+2]$ sejam ausentes, o resultado obtido será inválido. Com a pré imputação, um valor *dummy* preenche esta lacuna, validando portanto o resultado da operação.

Quanto melhor a estimativa deste valor *dummy*, melhor o resultado obtido para estimar o valor real de $x[d]$, considerando que a equação representada pelo indivíduo seja adequada o suficiente para regredir estes valores.

Posteriormente, faz-se uma análise em cada série temporal d da base de dados X_{VA} e, caso ela tenha valores ausentes, remove-se as instâncias com amostras em falta nesta série, de forma a gerar uma base de dados temporária X_{IM} que servirá como base de dados teste na programação genética. Durante o processo evolucionário, a aptidão de cada indivíduo é dada tendo como base a diferença entre as medidas média, variância e autocorrelação da série temporal imputada (d_{GP}) a partir da função de regressão que o indivíduo representa, e estas mesmas medidas correspondentes à série temporal original. Contudo, não é possível comparar d_{GP} à base de dados original, pois em uma aplicação real, não se sabe quais os valores das amostras ausentes. Por este motivo, optou-se por compara-la ao atributo correspondente em X_{IM}, d_{IM}, que contém toda a toda informação conhecida da série temporal original. A função de aptidão é descrita com mais detalhes na próxima seção.

Após a construção de X_{IM}, é possível executar a programação genética que comparará cada d_{GP} gerada pelos indivíduos com d_{IM}. Neste processo, primeiramente inicializa-se todos os indivíduos de forma aleatória. Cada um representa um função f que faz uso das informações contidas na base de dados pré-imputada X_{PI} para estimar os valores ausentes. Vale salientar que apenas os atributos não sendo regredidos são utilizados como parâmetro para a construção de cada função, portanto, se d é analisada, qualquer atributo exceto d pode ser utilizado como terminal. Depois que os indivíduos são avaliados, a PG entra no seu ciclo reprodutivo onde aplica-se as operações genéticas de cruzamento ou mutação e adiciona-se os indivíduos à próxima geração. Os detalhes desta etapa foram mencionados na seção 3.4.1.3.

O processo de avaliação e reprodução é repetido até que o número de gerações máxima seja atingido ou que o *fitness* seja igual a zero. Uma vez que este ciclo acabe, a função de regressão que representa o melhor indivíduo é retornada e usada para estimar as lacunas na série temporal d. Em seguida, a próxima série temporal na base de dados é analisada. Todo processo se repete até que todos os atributos sejam investigados e não haja mais valores ausentes. O pseudo-código do algoritmo é disponibilizado no Apêndice A.

4.1.1 Função de aptidão

O ponto chave de uma PG está na determinação da função de aptidão, também chamada de *fitness*. Os parâmetros escolhidos para compô-la foram baseados no trabalho de García, Kalenatic e Bello (2010), o qual utiliza dados estatísticos característicos da *função densidade de probabilidade* da distribuição que define a série temporal. Estes dados são a média, a variância e a autocorrelação entre as amostras e foram escolhidos pois estão diretamente relacionados às principais características de um série temporal: correlação entre amostras adjacentes, sazonalidades e a suavidade da variação entre amostras. A função de aptidão utilizada é descrita na Equação 4.14:

$$\mathcal{F} = |\mu_x - \mu_{x'}| + |\vartheta_x - \vartheta_{x'}| + \sum_{h=1}^{H} \frac{|\rho(h) - \rho'(h)|}{H}, \tag{4.14}$$

onde μ_x, ϑ_x e $\rho(h)$ são respectivamente a média, a variância e a autocorrelação da série temporal analisada d_{GP} e $\mu_{x'}$, $\vartheta_{x'}$ e $\rho'(h)$ são respectivamente a média, a variância e a autocorrelação dos dados da série temporal original conhecida d_{IM}.

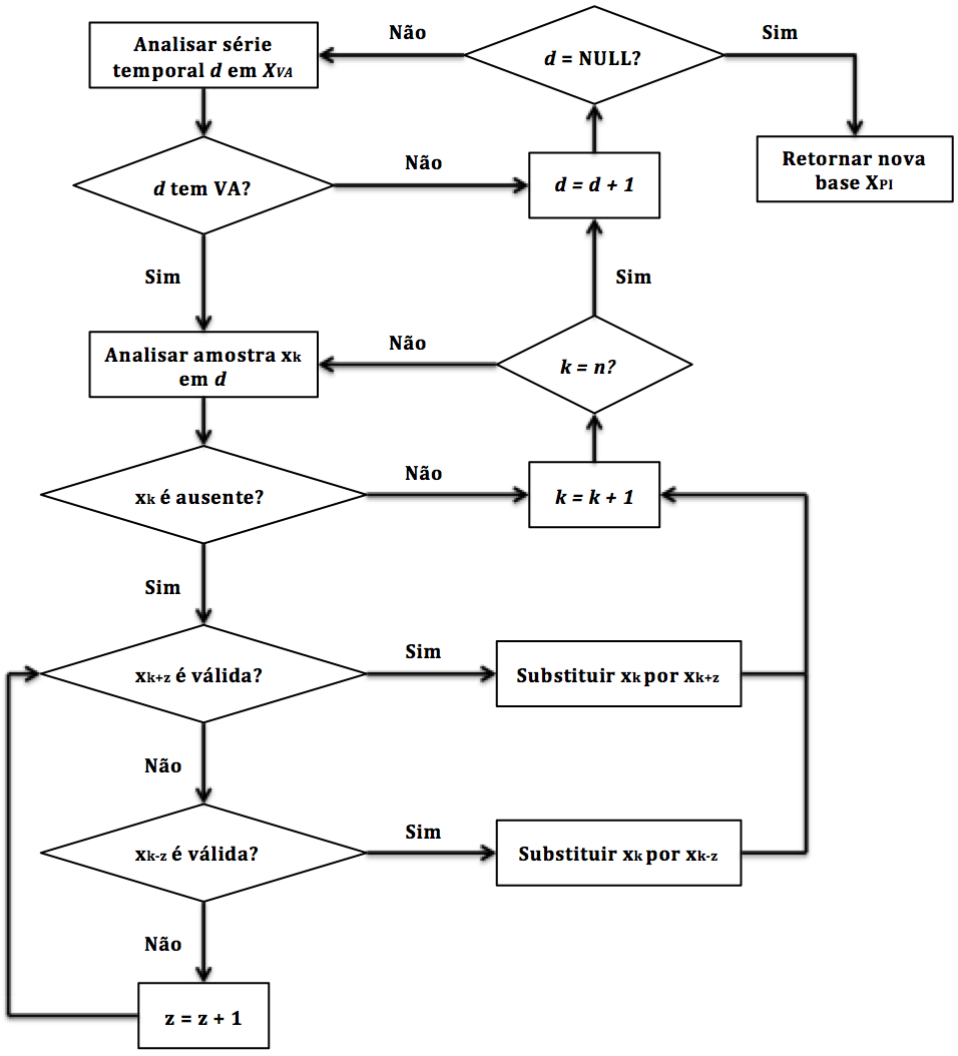
A partir da função de aptidão é possível saber o quão próximo do ótimo cada indivíduo está. No decorrer da PG, cada árvore é executada e sua aptidão ao problema é avaliada. A partir da Equação descrita em 4.14, percebe-se que o objetivo está relacionado à diferença entre os dados estatísticos da série construída a partir da eliminação das instancias com valores ausentes, e da série gerada a partir do indivíduo avaliado. Quanto menor for a diferença entre estes valores, mais apto o indivíduo está para reproduzir e ir para a próxima geração, ou seja, quanto menor o $\mathcal{F}$, maiores são suas chances de sobreviver.

4.2 O algoritmo GPImpute

Este algoritmo funciona conforme o fluxograma da Figura 4.11. Primeiramente lê-se a base de dados e mapeia-se os valores ausentes, depois faz-se uma pré-imputação dos dados e por fim, aplica-se a programação genética para gerar as funções de regressão de cada atributo com dados em falta. Estas funções descrevem unicamente a distribuição dos valores ausentes, podendo também ser aplicada ao resto da série temporal mas não com a mesma eficiência. Durante o processo de imputação serão construídas uma função de regressão para cada série temporal, e esta irá estimar todos os valores ausentes deste atributo.

A particularidade deste algoritmo está na forma que ele realiza a pré imputação dos dados. Este processo é descrito em detalhes no fluxograma da Figura 4.12. Nesta etapa do algoritmo, cada série temporal d da base de dados X_{VA} é analisada de forma separada, ou seja, os outros atributos da base são completamente ignorados. Em d verifica-se cada amostra x_k, caso esta seja ausente, busca-se pela amostra válida mais próxima a ela. Primeiramente são verificados os dados adjacentes à x_k, x_{k+z} e x_{k-z}, onde $z = 1$. Se ambos valores forem inválidos, z é incrementado em uma unidade e os próximos valores são verificados. Este processo ocorre até que um valor válido seja encontrado para substituir x_k. Ao final desta etapa, todos os atributos foram verificados e seus valores ausentes substituídos por valores vizinhos, formando então uma nova base X_{PI}.

Figura 4.12: Diagrama de fluxo da etapa de pré-imputação do GPImpute.



Fonte: Elaborada pela autora.

Esta abordagem foi escolhida por um simples motivo: séries temporais caracteristicamente tem pouca variação entre uma amostra e outra. Desta forma, ao substituir um valor ausente por seu vizinho, mantem-se essa variação mínima entre um dado e outro, uma abordagem a princípio melhor que substituir os dados em falta por valores aleatórios ou pela média da distribuição. Contudo, quando as lacunas são grandes, ou seja, quando há várias amostras ausentes adjacentes, gera-se um trecho constante, o que pode mudar consideravelmente a média da série temporal e também o desvio padrão. Por exemplo, se há 10 valores ausentes em sequência, todos estes 10 valores serão iguais a uma valor V, a média deste trecho será V e o desvio padrão será 0. Esta alteração estatística pode não afetar apenas este trecho, e sim a distribuição como um todo, comprometendo assim as características estatísticas da série temporal.

Para evitar este problema, deve-se substituir esta abordagem por um método que estime dados mantendo a propriedade que indica que há pouca variação entre amostras, mas de forma a não imputar valores iguais em todos os exemplos. Tendo isto em mente, optou-se por trocar a pré-imputação por amostras vizinhas pela pré-imputação por interpolação. O LGPImpute é o algoritmo que implementa isto e faz uso da interpolação de Lagrange para estimar os valores em falta.

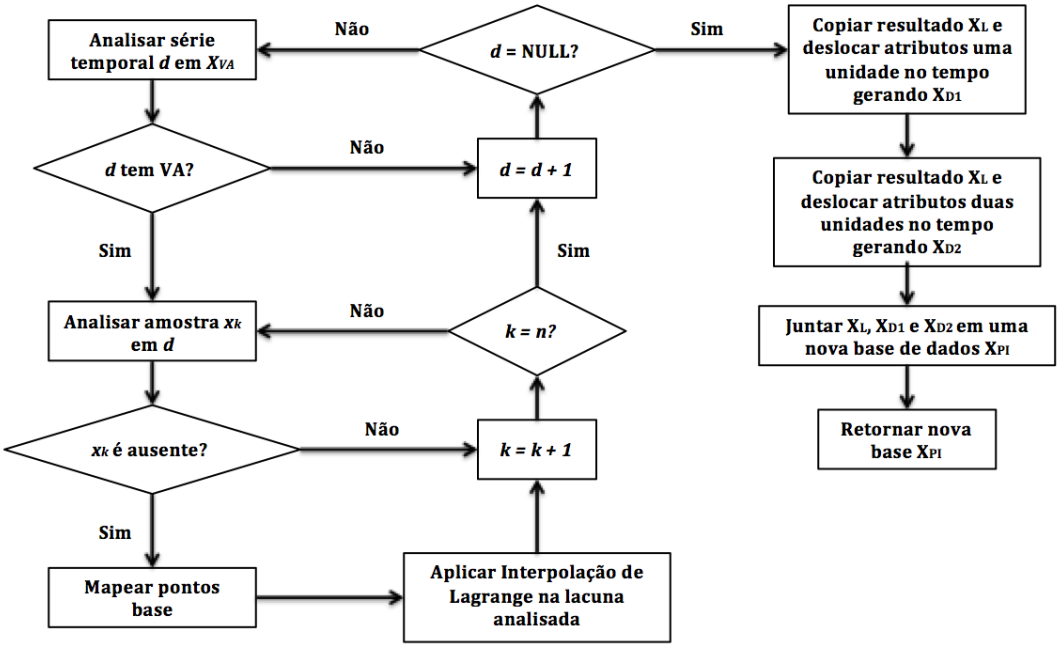
4.3 O algoritmo LGPImpute

O algoritmo LGPImpute difere do GPImpute em um aspecto apenas: a forma com a qual a pré imputação dos dados ocorre. Apesar da diferença ser em uma única etapa do algoritmo, esta mudança resulta em algumas implicações. Primeiro, as lacunas de cada série temporal são preenchidas por polinômios, o que garante variabilidade às amostras imputadas, respeitando a propriedade que indica que variações entre um exemplo e outro devem ser mínimas. Segundo, na pré-imputação do LGPImpute, a base de dados é triplicada e deslocada, o que possibilita que a PG faça uso de amostras passadas para estimar valores de amostras presentes, respeitando assim outra propriedade de séries temporais, o fato de amostras em um tempo $t-z$ influenciarem em amostras em um tempo t.

A Figura 4.13 ilustra o diagrama de fluxo da etapa de pré-imputação do LGPImpute. Assim como no GPImpute, cada série temporal d da base de dados X_{VA} é analisada de forma separada e verifica-se cada amostra x_k. Caso x_k seja ausente, mapeia-se os pontos base da interpolação para que seja construído o polinômio que definirá este ponto. Durante este pro-

cesso, se há uma sequência de valores ausentes, haverá apenas um polinômio que definirá esta sequência. O método para encontrar os pontos base e construir os polinômios foi desenvolvido neste trabalho e será descrito mais detalhadamente na próxima seção.

Figura 4.13: Diagrama de fluxo da etapa de pré-imputação do LGPImpute.



Fonte: Elaborada pela autora.

Uma vez que todas as amostras das lacunas da distribuição sejam estimadas por seus respectivos polinômios, a base de dados X_L resultante é copiada e deslocada em uma unidade, formando portanto a base de dados X_{D1}. Neste processo, se a base de dados X_L possui 3 séries temporais por exemplo, estes 3 atributos serão copiados e deslocados em uma unidade de tempo, ou seja, se antes a instância 1 começava do índice 0, esta agora começará do índice 1 e assim por diante. Ao final deste processo, a base de dados possuirá o dobro de atributos, logo, no exemplo dado, X_{D1} terá 6 séries temporais.

Como dito anteriormente, o motivo de dobrar os atributos da base de dados e deslocar a cópia deles, é o de permitir que a programação genética leve em consideração amostras passadas para estimar as amostras presentes. Desta forma, se um indivíduo representa a equação $x[d-1]+x[d+4]$, onde a base de dados possui apenas 3 atributos e $d=2$, para estimar os valores de cada amostra, será levado em consideração os valores das amostras do atributo 1 da série e do atributo 6. Porém, o atributo 6 é igual ao atributo 3 deslocado em uma unidade, ou seja, a amostra x_k do atributo 6 é na verdade a amostra x_{k-1} do atributo 3. Desta maneira, amostras

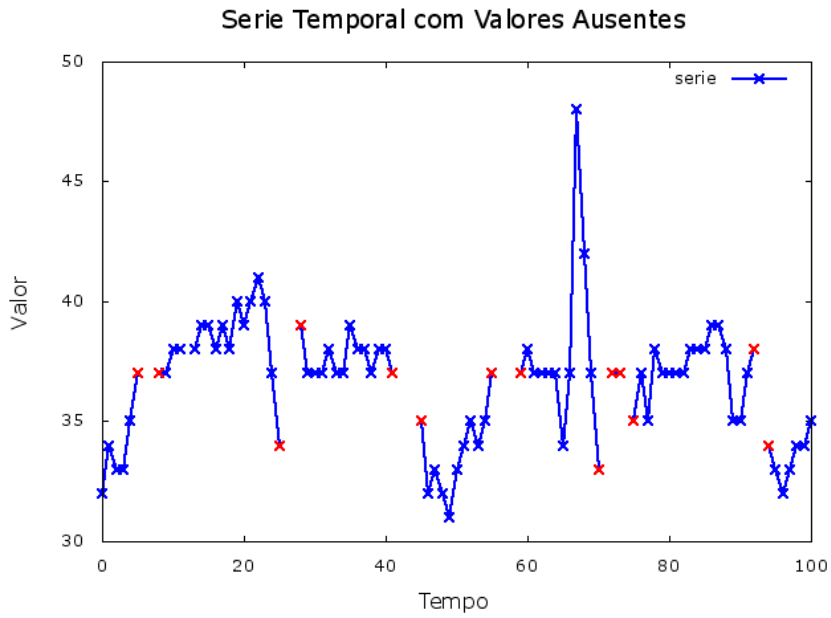
passadas podem ser consideradas para construir as funções de regressão.

Para aumentar a flexibilidade deste algoritmo, o conjunto de dados não é apenas duplicado, é triplicado, de forma que 3 grupos de dados compõem a base de dados resultante X_{PI}, são estes: X_L, que corresponde aos atributos imputados por interpolação de Lagrange; X_{D1} que consiste da cópia de X_L deslocada em uma unidade de tempo; e X_{D2}, que consiste de X_L deslocada em duas unidades de tempo. Assim, amostras em um tempo t, $t-1$ ou $t-2$ podem ser consideradas para estimar os valores ausentes na série temporal.

4.3.1 Determinação dos pontos base

Em uma série temporal com valores ausentes, os pontos usados como base na Fórmula de Lagrange são determinados a partir da vizinhança da lacuna na distribuição. O algoritmo interpolador irá procurar por todas as lacunas na série temporal e determinará seu respectivo tamanho s, o qual é definido pela quantidade de valores ausentes em sequência. A partir do tamanho s encontrado, o algoritmo ira procurar por 2 pontos conhecidos e vizinhos a uma distância s dos pontos extremos que delimitam a lacuna. Caso ambos pontos sejam encontrados, o polinômio interpolador terá 4 pontos base. Se 1 ponto for válido, o polinômio de Lagrange terá 3 pontos base. Por fim, se nenhum ponto vizinho a uma distância s dos extremos da lacuna for conhecido, apenas os extremos serão considerados e o polinômio terá 2 pontos base.

Figura 4.14: Exemplo de base de dados com valores ausentes.



Fonte: Elaborada pela autora.

Desta forma, se do tempo $t = 4$ ao tempo $t = 10$ os valores de $f(t)$ forem desconhecidos, a série temporal terá uma lacuna de tamanho $s = 7$. Os pontos extremos conhecidos dessa lacuna serão $t = 3$ como limite inferior e $t = 11$ como limite superior, e os pontos vizinhos aos extremos a uma distância s serão $t = -4$ e $t = 18$. Como $t = -4$ é um ponto inválido, ele será desconsiderado. Se $t = 18$ não for um valor ausente, este ponto será considerado, caso contrário, o polinômio interpolador terá apenas 2 pontos base, $t = 3$ e $t = 11$, os pontos correspondentes aos extremos da lacuna encontrada. Este processo é repetido até que todas as lacunas sejam analisadas e que seus valores sejam estimados. É gerado um polinômio por lacuna, mesmo que esta lacuna tenha apenas uma amostra ausente. A Figura 4.14 mostra um exemplo de base de dados com valores ausentes. Neste exemplo há 7 lacunas de diferentes tamanhos (número de amostras ausentes), e seus extremos estão identificados em vermelho.

5. Experimentos Computacionais

Para testar os métodos propostos, os algoritmos foram executados em 7 bases de dados distintas e foram julgados em termos de duas medidas de avaliação: RMSE e a diferença da autocorrelação entre a base original e a base imputada. As bases utilizadas são de aplicações reais e originalmente não possuem valores ausentes, estes foram introduzidos artificialmente em 3 diferentes proporções: 5% 10% e 30%. Ademais, para cada caso de teste, ambos algoritmos foram executados 10 vezes, pois por serem algoritmos estocásticos, os resultados podem variar consideravelmente. As próximas seções detalham os experimentos realizados e os resultados obtidos.

5.1 *Framework* Experimental

Como dito anteriormente, os experimentos foram conduzidos usando 7 bases de dados distintas, com três proporções de valores ausentes diferentes, 5%, 10% e 30%, totalizando 21 casos de teste. Cada caso testado foi executado 10 vezes tanto para o GPImpute quanto para o LGPImpute. As bases usadas estão especificadas na Tabela 5.2.

Cinco dos conjuntos de dados indicados na tabela foram escolhidos do repositório de Aprendizado de Máquina UCI (LICHMAN, 2013); *NN5 Complete 110* e *NNGC1 D1 V1 002* foram selecionadas do repositório KEEL (ALCALÁ-FDEZ A. FERNANDEZ, 2011). Nestas bases, os valores ausentes foram introduzidos artificialmente de forma completamente aleatória (MCAR). Além disso, os atributos não numéricos foram removidos e um novo atributo foi adicionado para indicar o tempo. Este atributo não possui dados faltosos.

A maioria das bases de dados usadas são originalmente multivariadas. As cinco primeiras bases indicadas na tabela referem-se a estudos relacionadas a análise do comportamento humano ou movimentos pertinentes a atividades físicas. Contudo, *Istanbul Stock*, *NN5 Complete 110* e *NNGC1 D1 V1 002* são bases ímpares. A primeira é uma base de dados com séries tem-

Base de dados	Acrônimo	Nº Atributos	Nº Instâncias	Aplicação
ADL Brush Teeth	ADBT	3	2167	Monitoramento
ADL Climb Stairs	ADCS	3	270	Monitoramento
ADL Comb Hair	ADCH	3	833	Monitoramento
Indoor Movement	IM	4	27	Monitoramento
Istanbul Stock	IS	10	536	Finanças
NN5 Complete 110	NN5	5	787	Finanças
NNGC1 D1 V1 002	NNG	5	1175	Monitoramento

Tabela 5.2: Base de dados usadas para testar os algoritmos GPImpute e LGPImpute.

porais univariadas e multivariadas que armazena informações a respeito dos índices da bolsa de valores de alguns países. Por outro lado, as duas últimas são ambas originadas de uma série temporal univariada onde os outros atributos correspondem a série temporal deslocada no tempo. *NN5 Complete 110* faz um estudo sobre os valores retirados de caixas eletrônicos no Reino Unido e *NNGC1 D1 V1 002* analisa o trafego de veículos em vias de trânsito.

5.1.1 Parametrização

Com o objetivo de encontrar os valores mais adequados para o número de indivíduos e número de gerações em ambos algoritmos desenvolvidos, foram testadas diferentes combinações desses parâmetros. Os valores 1024, 2048 e 4096 foram testados para o tamanho da população e os valores 64, 128 e 256 foram tetados para o número de gerações. Estes parâmetros foram então cruzados, totalizando 9 testes. Estes testes foram feitos nas 5 bases da Tabela 5.2 com menor número de instâncias.

Considerando a convergência do *fitness* da população de cada teste ao longo das gerações, e também os resultados obtidos em termos de RMSE e diferença da autocorrelação, notou-se que a melhor combinação de parâmetros para esta aplicação é de 2048 indivíduos e 128 gerações. Estes valores são usados tanto no GPImpute quanto no LGPImpute.

Outros parâmetros importantes ao codificar um PG são os operadores genéticos. A Tabela 5.3 mostra os parâmetros escolhidos para ambos algoritmos. As funções selecionadas são funções aritméticas, pois são operações simples e que geram resultados robustos. Considerar funções trigonométricas por exemplo, poderia introduzir alguma inconsistência na equação que o indivíduo representa, tal como $sen(exp(cos(x[3] + x[1])))$.

Os terminais são de dois tipos: constantes, definidas aleatoriamente e que podem assumir qualquer valor real; e variáveis, determinadas pelos valores das amostras dos atributos da base

Parâmetros	Valores
Conjunto de Funções	$+$, $-$, $\times$, $/$ (divisão protegida), abs
Conjunto de Terminais Variáveis	todos os atributo exceto o atributo sendo regredido
Conjunto de Terminais Constantes	valores aleatórios
Tamanho da população	2048
Inicialização	meio-a-meio
Número de Gerações	128
Probabilidade de cruzamento	75%
Probabilidade de recombinação	20%
Probabilidade de mutação	5%
Tipo de seleção	torneio (tamanho = 5)

Tabela 5.3: Parâmetros da Programação Genética usados no GPImpute e LGPImpute.

de dados não sendo regredidos. As probabilidades dos operadores de cruzamento, mutação e recombinação foram escolhidas de acordo com os valores recomendados por Eiben e Smith (2003) e Banzhaf et al. (1998), que seguem a convenção explicada na seção 3.4.1.3. Assim, como há 2048 indivíduos por população, a taxa de seleção escolhida é igual 20%; e como a recombinação deve ser priorizada, a taxa de mutação foi escolhida é igual a 5%.

5.1.2 Medidas de avaliação

Para avaliar os algoritmos propostos, estes foram comparados com três métodos de imputação disponíveis no software KEEL (ALCALÁ-FDEZ A. FERNANDEZ, 2011), EC, MC e KMI. Estes algoritmos foram escolhidos pois apesar de não serem as abordagens do estado da arte de imputação em séries temporais, representam três metodologias comumente utilizadas para imputar dados: imputação por clusterização (EC e KMI), imputação baseada na distribuição dos dados (MC) e imputação baseada nos vizinhos mais próximos (KMI). O KMI foi executado com $k = 5$ e o EC com os parâmetros recomendados por (LUENGO, GARCÍA; HERRERA, 2012).

As medidas de avaliação foram duas: RMSE, que indica o quão distante do valor real o valor imputado está; e a variação da auto-correlação a um atraso de 5 amostas. Esta variação é determinada pelo módulo da diferença da auto-correlação da base original e da base imputada. Como a auto-correlação varia de uma série temporal para outra, este valor na base de dados como um todo é definido como a soma do resultado obtido para cada atributo, exceto o atributo que indica o tempo, dividido pelo número de atributos menos um. As Equações 5.15 e 5.16

correspondem ao cálculo do RMSE e variação de auto-correlação respectivamente.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \hat{x_i})^2} \tag{5.15}$$

onde n é o número de amostras ausentes na série temporal, x_i é o valor da amostra na base de dados original e $\hat{x_i}$ é o valor da amostra imputado.

$$DIFF = |\frac{1}{d}\sum_{i=1}^{d}\rho(h)_i - \frac{1}{d}\sum_{i=1}^{d}\hat{\rho}(h)_i|, \tag{5.16}$$

onde d é o número de séries temporais na base de dados, h é o atraso, $\rho(h)$ é a auto-correlação da série temporal da base original e $\hat{\rho}(h)$ é a auto-correlação da série temporal da base imputada. Desta forma, quanto menor o valor de $DIFF$ melhor a performance do algoritmo em termos de manter as características originais da base. Vale salientar que d começa do atributo 1 pois o atributo 0 corresponde ao índice do tempo.

5.2 Resultados

Os algoritmos GPImpute e LGPImpute geram como resultado uma função de regressão para cada série temporal analisada, e geram também a base de dados imputada. A partir das funções de regressão, pode-se fazer um estudo do comportamento dos valores ausentes na distribuição da série temporal. A partir da base de dados, pode-se avaliar o quão eficaz é a estimação dos valores e se as propriedades originais da distribuição foram mantidas. As próximas seções abordam cada uma destas análises.

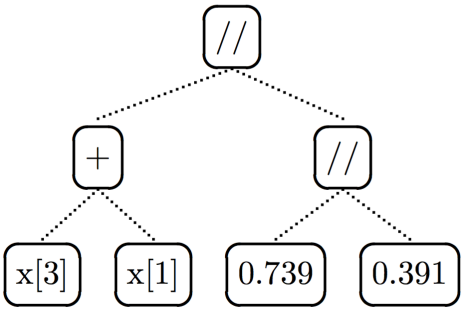
5.2.1 Funções de regressão obtidas

Uma das principais vantagens dos algoritmos desenvolvidos é o fato deles gerarem modelos regressivos compreensíveis que podem caracterizar precisamente um conjunto de dados. Além disso, ambos são capazes de encontrar possíveis interrelações entre diferentes atributos em bases de dados multivariadas, e estas interrelações podem ser extraídas a partir da análises das funções de regressão resultantes. Neste trabalho, apenas duas funções são estudas, uma resultante do GPImpute e outra do LGPImpute.

A Figura 5.15 ilustra o melhor indivíduo encontrado pelo GPImpute para atributo 2 da

base *ADL Comb Hair*. Nota-se que para construir a função de regressão desta série temporal, foram usados os atributos $x[1]_t$ e $x[3]_t$, que consistem de outras séries temporais. Pode-se dizer portanto que, cada valor ausente no atributo 2 é igual à $\frac{x[3]_t+x[1]_t}{0.739/0.391}$, ou seja, $x[2]_t = \frac{x[3]_t+x[1]_t}{0.739} \times 0.391$ em um tempo t, onde os valores entre colchetes indicam uma série temporal d na base de dados.

Figura 5.15: Árvore de regressão do atributo 2 da base ADL Comb Hair.



Fonte: Elaborada pela autora.

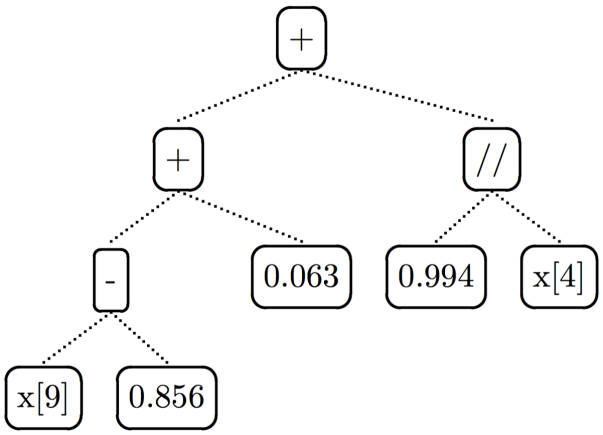
Em contrapartida, a Figura 5.16 mostra o melhor indivíduo encontrado pelo LGPImpute para o atributo 3 da base *ADL Comb Hair*. Neste caso a função de regressão depende dos atributos 4 e 9. Porém, como indicado na tabela 5.2, esta base de dados possui apenas 3 atributos. O atributo 9 é considerado pois a base foi triplicada, de forma que esta série temporal representa de fato a série temporal do atributo 3 deslocada em duas unidades no tempo. Pode-se dizer então que, de acordo com a equação representada pela Figura 5.16, os valores ausentes do atributo 3 podem ser estimados tendo como base as amostras do próprio atributo duas unidades de tempo antes. Em outras palavras, $x[3]_t = x[9]_t - 0.856 + 0.063 + \frac{0.994}{x[4]_t}$, onde $x[9] = x[3]_{t-2}$.

Para analisar o comportamento destas funções, pode-se esboçar o gráfico de cada uma e compará-los com os gráficos das distribuições originais, analisando também se a correlação encontrada entre os atributos é condizente com as características da aplicação em questão, que no caso da base *ADL Comb Hair*, refere-se ao movimento de um braço ao pentear o cabelo. Contudo, como neste trabalho o foco principal é investigar a eficiência da imputação dos dados, a análise do comportamento destas funções não é feita.

5.2.2 Avaliação da imputação para o RMSE

Um importante aspecto a ser analisado ao aplicar um modelo de imputação em uma base de dados, é a precisão da predição das amostras ausentes. Esta precisão é computada a partir do

Figura 5.16: Árvore de regressão do atributo 3 da base ADL Comb Hair.



Fonte: Elaborada pela autora.

RMSE, medida de avaliação indicada na Equação 5.15. Com o objetivo de avaliar a precisão da imputação em diferentes proporções de valores ausentes, cada grupo de bases de teste foi analisado separadamente. Ademais, tendo como base os resultados obtidos pelo RMSE, aplicou-se tambéd o teste estatístico de *Friedman*, que indica melhor a superioridade de um algoritmo em relação ao outro, pois ranqueia os algoritmos comparados considerando todos os resultados obtidos.

As Tabelas 5.4, 5.5 e 5.6 mostram os resultados obtidos para a medidas de avaliação RMSE em grupos de bases de dados com 5%, 10% e 30% de valores ausentes respectivamente. Vale salientar que, para deixar os resultados mais claros, os dados foram normalizados de modo que o valor 1 indica o melhor caso e o valor 0 o pior. São indicados os resultados obtidos para os algoritmos EC, KMI, MC, GPImpute e LGPImpute. Para os dois últimos, como ambos foram executados 10 vezes, são mostrados nas tabelas os valores referentes ao melhor resultado de cada algoritmo, GPI_ρ e $LGPI_\rho$, e a média dos 10 resultados obtidos para cada um, GPI_μ e $LGPI_\mu$.

O Teste de Friedman foi utilizado para avaliar estatisticamente os resultados. Este é um teste estatístico não-paramétrico que melhor demonstra a diferença de performance de cada método julgado. Ele ranqueia as abordagens avaliadas classificando cada linha da tabela, e posteriormente considera os valores dos ranques por coluna, indicando portanto, a partir de uma análise global, o quão bom o método avaliado é em comparação as outras abordagens estudadas. Os resultados deste teste estão dispostos na última linha das Tabelas 5.4, 5.5 e 5.6, de forma que quanto menor o valor do ranque, melhor é o algoritmo.

Nas Tabelas 5.4, 5.5 e 5.6 são também indicadas quais hipóteses são consideradas estatisticamente relevantes para o conjunto de testes realizados. Esta análise é feita com de acordo com o procedimento de ***Holm/Shaffer*** a um nível de significância α de $0,10$. Este procedimento é aplicado com o intuito de descartar as hipóteses falso-positivas. Nas tabelas $\{A\} \succ \{B, C\}$ indica que o método A é estatisticamente superior aos métodos B e C. Os detalhes dos resultados obtidos por este procedimento estão indicados no Apêndice B.

Base de Dados	EC	KMI	MC	GPI_ρ	GPI_μ	$LGPI_\rho$	$LGPI_\mu$
ADBT	0,671	0,877	0,394	0,237	0,000	**1,000**	0,547
ADCS	0,077	0,357	0,227	0,192	0,000	**1,000**	0,916
ADCH	0,311	0,468	0,000	0,417	0,064	**1,000**	0,932
IM	0,000	**1,000**	0,657	0,795	0,689	0,734	0,726
IS	0,838	**1,000**	0,991	0,352	0,000	0,310	0,084
NN5	0,388	0,813	0,000	**1,000**	0,892	0,902	0,815
NNG	0,167	**1,000**	0,000	0,905	0,834	0,959	0,886
F. Rank	**5,141**	**2,286**	**5,428**	**3,571**	**5,714**	**2,142**	**3,714**
$\{KMI\} \succ \{MC, GPI_\mu\}, \{LGPI_\rho\} \succ \{MC, GPI_\mu\}$							

Tabela 5.4: Resultado do NRMSE e do rank de Friedman para o GPImpute e LGPImpute em bases de dados com 5% de valores ausentes.

Base de Dados	EC	KMI	MC	GPI_ρ	GPI_μ	$LGPI_\rho$	$LGPI_\mu$
ADBT	0,762	**1,000**	0,388	0,350	0,000	0,616	0,235
ADCS	0,084	0,105	0,218	0,172	0,000	**1,000**	0,757
ADCH	0,293	0,578	0,020	0,576	0,000	**1,000**	0,884
IM	0,662	**1,000**	0,000	0,333	0,264	0,665	0,315
IS	0,856	0,999	**1,000**	0,595	0,347	0,720	0,000
NN5	0,343	0,827	0,000	0,829	0,755	**1,000**	0,927
NNG	0,000	0,541	0,137	0,963	0,854	**1,000**	0,853
F. Rank	**4,571**	**2,999**	**4,857**	**3,857**	**5,857**	**1,857**	**3,999**
$\{LGPI_\rho\} \succ \{GPI_\mu\}$							

Tabela 5.5: Resultado do NRMSE e do rank de Friedman para o GPImpute e LGPImpute em bases de dados com 10% de valores ausentes.

Ao analisar os resultados do teste de *Friedman*, percebe-se que para todas as proporções de valores ausentes estudadas, o LGP_ρ é melhor que os algoritmos comparados. Além disso, tendo como base o procedimento de ***Holm/Shaffer***, pode-se concluir que este método é estatisticamente superior aos métodos MC e GP_ρ em bases de dados com 5% de dados em falta, e é superior ao GP_ρ em bases com 10% e 30% de valores ausentes.

Contudo, o mesmo não pode ser dito a respeito do LGP_μ, que é referente a média dos 10 resultados obtidos para o LGPImpute. De acordo com o ranque de *Friedman*, este é melhor

Base de Dados	EC	KMI	MC	GPI_{ρ}	GPI_{μ}	$LGPI_{\rho}$	$LGPI_{\mu}$
ADBT	0,769	**1,000**	0,317	0,104	0,000	0,908	0,527
ADCS	0,140	0,400	0,394	0,023	0,000	**1,000**	0,871
ADCH	0,350	0,517	0,052	0,537	0,000	**1,000**	0,876
IM	0,932	**1,000**	0,893	0,901	0,000	0,925	0,891
IS	0,952	**1,000**	0,981	0,740	0,000	0,778	0,643
NN5	0,249	0,787	0,000	**1,000**	0,852	0,858	0,802
NNG	0,000	0,793	0,215	0,842	0,787	**1,000**	0,900
F. Rank	**4,428**	**2,714**	**5,000**	**4,000**	**6,142**	**1,999**	**3,714**
$\{KMI\} \succ \{GPI_{\mu}\}, \{LGPI_{min}\} \succ \{GPI_{\mu}\}$							

Tabela 5.6: Resultado do NRMSE e do rank de Friedman para o GPImpute e LGPImpute em bases de dados com 30% de valores ausentes.

que EC, MC e GP_{μ} em todos os grupos de bases de dados testadas, perdendo unicamente para o KMI. Este resultado evidência a característica estocástica deste método que, por usar programação genética, não é capaz de *sempre* determinar o melhor resultado. Outro exemplo deste fenômeno é disposto no Apêndice C. Neste artigo, os resultados foram obtidos a partir de 10 execuções diferentes das usadas no presente trabalho, e eles divergem consideravelmente dos resultados mostrados neste capítulo. Devido a esta grande variação de performance, recomenda-se que o algoritmo seja executado pelo menos 20 vezes para assegurar os resultados.

Em relação à performance do GPImpute, de acordo com os resultados apontados nas Tabelas 5.4, 5.5 e 5.6, pode-se concluir que este é inferior ao LGPImpute. Ao considerar somente os resultados obtidos a partir da média dos testes por exemplo, percebe-se que para todos os grupos de dados testados, o algoritmo é superado pelos outros métodos comparados. Nota-se também que a performance do GPImpute decai a medida que o número de valores ausentes aumenta na base de dados. Isso se dá devido a forma com a qual a pré-imputação dos dados foi efetuada, que substitui os dados em falta por amostras vizinhas. Porém, quando a lacuna a ser preenchida é muito grande, esta abordagem adiciona um trecho constante na série temporal e interfere consideravelmente na distribuição dos dados.

Outro aspecto que coloca o GPImpute em desvantagem em relação ao LGPImpute é o fato dele não usar os valores de amostras passadas para determinar as amostras presentes, uma característica importante a ser considerada ao lidar com séries temporais, pois as amostras na distribuição dos dados estão fortemente correlacionadas. Assim, tendo como base estes resultados, conclui-se que o LGPImpute é mais eficiente que o GPImpute, EC e MC e pode ser mais eficiente que o KMI. Ademais, é importante frisar que tanto o LGPImpute quanto o GPImpute produzem resultados variados, característica inerente da programação genética. Todavia, no

caso do LGPImpute, resultados satisfatórios podem ser facilmente obtidos.

5.2.3 Avaliação da imputação para o DIFF

A Tabela 5.7 indica a diferença absoluta obtida entre a auto-correlação da base de dados imputada e a base de dados original. Como dito anteriormente, esta avaliação tem por objetivo mostrar a magnitude da distorção das propriedades originais da série temporal causada pela imputação de dados, logo, quanto menor o valor da diferença encontrada, melhor o método é em termos de manter as características estatísticas originais da base. Nesta tabela os melhores resultados estão destacados em negrito.

Base	MV	EC	KMI	MC	GPI_ρ	GPI_μ	$LGPI_\rho$	$LGPI_\mu$
ADBT	5	2.094	2.644	2.788	3.048	3.418	**1.085**	1.598
	10	**3.510**	4.174	4.047	4.924	5.701	3.617	4.097
	30	10.742	11.032	11.534	7.358	10.398	**2.255**	5.014
ADCS	5	0.061	0.128	0.194	0.040	0.171	**0.020**	0.037
	10	0.283	0.328	0.374	0.011	0.189	**0.015**	0.091
	30	0.926	1.046	0.642	0.145	0.377	**0.032**	0.180
ADCH	5	2.563	3.901	7.859	2.252	6.586	**0.297**	0.522
	10	6.265	8.929	15.430	4.218	15.250	**0.019**	0.517
	30	19.220	27.231	41.254	8.217	38.455	**0.528**	1.945
IM	5	**0.000**	**0.000**	0.001	0.002	0.002	0.002	0.002
	10	**0.000**	**0.000**	**0.000**	0.001	0.002	0.001	0.002
	30	0.003	0.002	**0.000**	0.002	0.044	0.001	0.004
IS	5	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	10	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	30	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
NN5	5	4.623	3.749	5.621	0.070	0.187	**0.038**	0.423
	10	9.116	8.715	10.939	0.152	0.328	**0.057**	0.714
	30	29.559	16.872	30.909	0.699	2.492	**0.672**	2.038
NNG	5	2614.366	1467.156	3808.596	**12.830**	153.591	64.166	199.769
	10	5538.947	2981.819	6566.467	**10.157**	147.786	69.603	355.502
	30	17765.530	1686.537	18316.717	**18.969**	521.754	31.627	208.270

Tabela 5.7: Resultado da variação de auto-correlação para os algoritmos GPImpute e LGPImpute.

É possível perceber a partir destes resultados que novamente o LGP_ρ se destaca em relação aos métodos comparados. Mesma ao analisar o LGP_μ, de forma geral, o LGPImpute é superior aos métodos GPImpute, EC, MC e KMI no que diz respeito manter as propriedades originais da distribuição dos dados após a imputação.

Esta é uma característica extremamente importante, pois em muitas aplicações de séries temporais é vital que os dados estatísticos não sejam alterados uma vez que estes dados são

utilizados em modelos preditivos e de extração de conhecimento, logo, quanto menor a distorção estatística provocada sobre a distribuição, melhor o método de imputação.

No caso do GPImpute, na maioria dos casos testados a distorção inserida na base de dados foi menor que nos métodos EC, MC e KMI, sendo que para a base *NNGC1 D1 V1 002*, este algoritmo obteve resultados melhores que o LGPImpute. O fato de ambos métodos desenvolvidos responderem bem à esta medida de avaliação indica que a função de aptidão escolhida é adequada. Isto indica também que os algoritmos respondem bem a esta função e conseguem alcançar o objetivo imposto por ela.

6. Considerações Finais

Neste trabalho, dois algoritmos de imputação de dados baseados em programação genética foram propostos, chamados de GPImpute e LGPImpute. Estes algoritmos visam estimar valores ausentem em bases de dados de séries temporais multivariadas de forma a manter as propriedades estatísticas originais da distribuição e fundamentam-se em duas premissas: (i) séries temporais podem ser representadas por um processo estocástico onde características como média, variância e auto-correlação são estatísticas vitais a serem consideradas ao tratar valores ausentes para manter as propriedades originais da estrutura; e (ii) a programação genética é capaz de construir funções de regressão a partir de dados de exemplo eficientemente. Como resultado, os algoritmos desenvolvidos produzem modelos compreensíveis que além de imputar dados podem também ser usados para análise de padrões.

O GPImpute e LGPImpute foram testados em 7 bases de dados de diferentes áreas de aplicação, tamanhos e proporções de valores ausentes. Ademais, eles foram comparados com três métodos bem conhecidos de imputação de dados disponibilizados no *software* KEEL: KMI, EC e MC. Os algoritmos de comparação foram escolhidos pois representam três abordagens comumente utilizadas para estimar valores ausentes em uma base de dados, imputação estatística (MC), imputação por clusterização (KMI e EC) e imputação baseada nos vizinhos mais próximos (KMI). Além disso, cada caso de teste foi executado 10 vezes para cada método desenvolvido.

A maior vantagem do GPImpute e LGPImpute é o fato destes proporcionam um modelo compreensível ao final do processo e considerarem interrelações entre séries temporais multivariadas, o que permite que analistas de dados possam compreender melhor as características da ausência de dados, e possam extrair padrões indicados pela função de regressão resultante.

O primeiro algoritmo proposto não obteve bons resultados em termos da medida de avaliação RMSE, porém este foi capaz de manter as propriedade originais da distribuição dos dados após a imputação, obtendo resultados satisfatórios quando comparadas a autocorrelação

da base imputada com a da base de dados original. No caso do LGPImpute os resultados foram mais satisfatórios. Quando analisados todos os casos de teste, este algoritmo mostrou-se eficiente para ambas as medidas de avaliação estudadas, superando os métodos comparados quando avaliada a capacidade de manter as propriedades originais da série temporal, e superando os métodos EC, MC e GPImpute e termos de acurácia de predição. Porém, quando analisados os melhores resultados das 10 execuções feitas, o cenário altera e o LGPImpute passa a superar todos os métodos de comparação. Esta variação nos resultados evidencia as características estocásticas da programação genética, e mostra também que são necessárias mais de uma execução para se obter resultados robustos.

Apesar da baixa performance em termos de RMSE do GPImpute, este método se mostrou promissor servindo como base para o LGPImpute. Este último algoritmo se mostrou bem melhor que a primeira versão desenvolvida, sendo este capaz de manter as características originais das bases de dados de forma superior aos métodos EC, MC e KMI e obtendo resultados bastante satisfatórios para todas as proporções de valores ausentes avaliadas, sendo assim uma alternativa viável para a imputação de dados em séries temporais.

6.1 Dificuldades Encontradas

As maiores dificuldades encontradas durante o desenvolvimento deste trabalho estão relacionadas à implementação da programação genética. O principal problema enfrentado foi a codificação da classe de variáveis, a qual deveria ser um vetor que representa o valor de um exemplo em cada atributo. Esta dificuldade foi sanada com a ajuda do autor do artigo Tran, Zhang e Andreae (2015), Cao Truong Tran, que compartilhou sua implementação. Outro problema confrontado foi a implementação do método responsável pela definição da aptidão do indivíduo dentro do ECJ, que deveria comunicar-se de alguma forma com a base de dados de teste e a base de dados avaliada. Para solucionar este problema, foi criado um objeto na memória estática da máquina virtual que representasse a base de dados e pode ser acessado rapidamente por todos os métodos durante a execução do programa.

Foram enfrentados também problemas de ordem conceitual em relação à forma com a qual uma série temporal é caracterizada e os aspectos mais importantes a serem considerados, como também problemas conceituais em relação a maneira com a qual a programação genética se comporta. Estas questões foram resolvidas a partir de leituras aprofundadas e detalhadas a

respeito destes temas.

6.2 Trabalhos Futuros

Para aprimorar os resultados obtidos, em trabalhos futuros pode-se tornar o LGPImpute uma abordagem híbrida, aplicando antes dele outros métodos computacionais que extraiam características da base de dados. Métodos como Redes Neurais e Redes Bayesianas por exemplo, são capazes de encontrar interrelações entre atributos diferentes, a partir disto é possível então introduzir este conhecimento na função de aptidão da programação genética sanando portanto o problema da função escolhida não caracterizar bem a distribuição dos dados da série temporal. É possível também usar métodos de controle de parâmetros, que consiste na implementação de operadores genéticos adaptativos que se modificam dependendo do estágio evolutivo que se encontram, podendo assim aumentar a performance do algoritmo evolucionário.

Uma abordagem interessante seria gerar uma função de regressão para determinados trechos da série temporal, isto provavelmente melhoraria a acurácia de predição do algoritmo uma vez que o espaço de busca diminuiria. Estes trechos seriam determinados de acordo com a varições marcantes na distribuição dos dados e seriam analisados separadamente pela programação genética, de forma a melhorar a precisão das funções de regressão encontradas.

Por fim, Gosavi (2014) sugere que se utilize algoritmos de aprendizado reforçado para criar modelos estocásticos de otimização, este pode também ser utilizado de forma híbrida com o LGPImpute. Como séries temporais são definidas como processos estocásticos e a imputação de dados é um problema de otimização, esta abordagem pode ser bastante eficiente para contornar os problemas enfrentados pelo algoritmo proposto, podendo inclusive lidar melhor com estruturas estocásticas não estacionárias.

Referencias Bibliográficas

ALCALÁ-FDEZ A. FERNANDEZ, J. L. J. D. S. G. L. S. F. H. J. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, v. 17:2-3, p. 255–287, 2011. Citado 2 vezes nas páginas 33 e 35.

AYDILEK, I. B., ARSLAN, A. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*, Elsevier, v. 233, p. 25–35, 2013. Citado na página 6.

BANZHAF, W. et al. Genetic programming—an introduction: On the automatic evolution of computer programs and its applications, dpunkt. verlag and morgan kaufmann publishers. *Inc., San Francisco, California*, 1998. Citado 2 vezes nas páginas 19 e 28.

BARNARD, J., MENG, X.-L. Applications of multiple imputation in medical studies: from aids to nhanes. *Statistical Methods in Medical Research*, SAGE Publications, v. 8, n. 1, p. 17–36, 1999. Citado na página 12.

BATISTA, G. E., MONARD, M. C. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, Taylor & Francis, v. 17, n. 5-6, p. 519–533, 2003. Citado na página 13.

BATISTA, G. E., MONARD, M. C. et al. A study of k-nearest neighbour as an imputation method. *HIS*, v. 87, n. 251-260, p. 48, 2002. Citado na página 5.

BERRUT, J.-P., TREFETHEN, L. N. Barycentric lagrange interpolation. *Siam Review*, SIAM, v. 46, n. 3, p. 501–517, 2004. Citado 2 vezes nas páginas 21 e 22.

BOX, G., JENKINS, G. M., REINSEL, G., *Time Series Analysis: Forecasting and Control*. [S.l.]: John Wiley & Sons, 2008. Citado 6 vezes nas páginas 1, 5, 12, 14, 15 e 16.

BRUNO, B. et al. Analysis of human behavior recognition algorithms based on acceleration

data. Em: IEEE. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. [S.l.], 2013. p. 1602–1607. Citado na página 1.

CAI, Y. et al. Facets: Fast comprehensive mining of coevolving high-order time series. Em: ACM. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.], 2015. p. 79–88. Citado 3 vezes nas páginas 6, 15 e 16.

DEMPSTER, A. P., LAIRD, N. M., RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, JSTOR, p. 1–38, 1977. Citado na página 5.

EIBEN, A. E., SMITH, J. E. Introduction to evolutionary computing. [S.l.]: *Springer, 2003*. v. 53. Citado 6 vezes nas páginas 17, 18, 19, 25, 28 e 30.

FAYYAD, U., PIATETSKY-SHAPIRO, G., SMYTH, P. From data mining to knowledge discovery in databases. *AI magazine*, 1996. Citado 2 vezes nas páginas 9 e 11.

FRANCO, N. B. Cálculo numérico. [S.l.]: *Pearson, 2006*. Citado na página 21.

GARCÍA, J. C. F., KALENATIC, D., BELLO, C. A. L. An evolutionary approach for imputing missing data in time series. *Journal of Circuits, Systems, and Computers, World Scientific*, v. 19, n. 01, p. 107–121, 2010. Citado 6 vezes nas páginas 5, 6, 15, 16, 27 e 41.

GARCÍA, J. C. F., KALENATIC, D., BELLO, C. A. L. Missing data imputation in multivariate data by evolutionary algorithms. *Computers in Human Behavior, Elsevier*, v. 27, n. 5, p. 1468–1474, 2011. Citado na página 5.

GARCÍA-LAENCINA, P. J., SANCHO-GÓMEZ, J.-L., FIGUEIRAS-VIDAL, A. R. Pattern classification with missing data: a review. Neural Computing and Applications, *Springer*, v. 19, n. 2, p. 263–282, 2010. Citado na página 5.

GAUTAM, C., RAVI, V. Data imputation via evolutionary computation, clustering and a neural network. Neurocomputing, *Elsevier*, v. 156, p. 134–142, 2015. Citado na página 6.

GHEYAS, I. A., SMITH, L. S. A neural network-based framework for the reconstruction of incomplete data sets. *Neurocomputing, Elsevier*, v. 73, n. 16, p. 3039–3065, 2010. Citado na página 12.

GOSAVI, A. Simulation-based optimization: parametric optimization techniques and reinforcement learning. [S.l.]: *Springer, 2014*. v. 55. Citado na página 45.

GRAHAM, J. W. Missing data analysis: Making it work in the real world. *Annual review of psychology, Annual Reviews*, v. 60, p. 549–576, 2009. Citado 2 vezes nas páginas 5 e 12.

HALL, M. et al. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 1018, 2009. Citado na página 1.

HAN, J., KAMBER, M., PEI, J., Data mining: concepts and techniques. [S.l.]: *Elsevier, 2011*. Citado na página 9.

HONAKER, J., KING, G. What to do about missing values in time-series cross-section data. *American Journal of Political Science, Wiley Online Library*, v. 54, n. 2, p. 561–581, 2010. Citado 2 vezes nas páginas 1 e 2.

HRUSCHKA, E. R. et al. A survey of evolutionary algorithms for clustering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, IEEE*, v. 39, n. 2, p. 133–155, 2009. Citado na página 5.

HRUSCHKA, E. R. et al. On the influence of imputation in classification: practical issues. *Journal of Experimental & Theoretical Artificial Intelligence, Taylor & Francis*, v. 21, n. 1, p. 43–58, 2009. Citado 3 vezes nas páginas 2, 5 e 12.

JUNGER, W., LEON, A. P. de. Imputation of missing data in time series for air pollutants. Atmospheric Environment, *Elsevier*, p. 96–104, 2015. Citado 2 vezes nas páginas 1 e 6.

KELLY, J. Big data vendor revenue and market forecast. *Wikibon Article. Febrero, 2014*. Citado 2 vezes nas página 9 e 10.

KOZA, J. R. Genetic programming: on the programming of computers by means of natural selection. [S.l.]: *MIT press, 1992*. v. 1. Citado 2 vezes nas páginas 17 e 19.

LAGRANGE, *J. L. Leçons élémentaires sur les Mathématiques, données à l'École normale, en 1795*. [S.l.: s.n.], 1812. Citado na página 21.

LEKE, C., MARWALA, T., PAUL, S. Proposition of a theoretical model for missing data imputation using deep learning and evolutionary algorithms. *ArXiv preprint arXiv: 1512.01362, 2015*. Citado na página 6.

LI, D. et al. Towards missing data imputation: a study of fuzzy k-means clustering method. Em: *SPRINGER. Rough sets and current trends in computing*. [S.l.], 2004. p. 573–579. Citado na página 14.

LICHMAN, *M. UCI Machine Learning Repository*. 2013. Disponível em: (http: //archive.ics.uci.edu/ml). Citado na página 33.

LINGRAS, P., ZHONG, M., SHARMA, S. Evolutionary regression and neural imputations of missing values. Em: *Soft Computing Applications in Industry*. [S.l.]: *Springer, 2008*. p. 151–163. Citado na página 5.

LITTLE, R., RUBIN, D. Analysis with missing data. [S.l.]: *John Wiley & Sons, New York, 1987*. Citado 2 vezes nas páginas 1 e 13.

LOBATO, F. et al. Multi-objective genetic algorithm for missing data imputation. *Pattern Recognition Letters, Elsevier*, v. 68, p. 126–131, 2015. Citado 2 vezes nas páginas 1 e 6.

LUENGO, J., GARCÍA, S., HERRERA, F. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems, Springer*, v. 32, n. 1, p. 77–108, 2012. Citado 3 vezes nas páginas 2, 12 e 35.

LUKE, S. et al. A java-based evolutionary computation research system. Online (March 2004) http://cs.gmu.edu/ẽclab/projects/ecj, 2004. Citado 3 vezes nas páginas 2, 13 e 27.

PAN, R. et al. Missing data imputation by k nearest neighbours based on grey relational structure and mutual information. *Applied Intelligence, Springer*, v. 43, n. 3, p. 614–632, 2015. Citado na página 5.

POLI, R. et al. A field guide to genetic programming. [S.l.]: *Lulu. com, 2008*. Citado 7 vezes nas páginas 17, 18, 19, 20, 24, 25 e 26.

SHUMWAY, R. H., STOFFER, D. S. Time series analysis and its applications: with R examples. [S.l.]: *Springer Science & Business Media, 2010*. Citado 3 vezes nas páginas 1, 15 e 16.

TRAN, C. T., ZHANG, M., ANDREAE, P. Multiple imputation for missing data using genetic programming. Em: *ACM. Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*. [S.l.], 2015. p. 583–590. Citado 4 vezes nas páginas 2, 6, 17 e 44.

WONG, A. K., CHIU, D. K. Synthesizing statistical knowledge from incomplete mixed-mode data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, IEEE*, n. 6, p. 796–805, 1987. Citado 2 vezes nas página 10 e 13.

ZAKI, M. J., JR, W. M. Data mining and analysis: fundamental concepts and algorithms. [S.l.]: *Cambridge University Press, 2014*. Citado na página 11.

Apêndice A - Pseudo-código da Imputação

O Algoritmo 2 descreve o pseudo-código dos métodos codificados. Este algoritmo seque a mesma linha de raciocínio indicada no diagrama 4.11 explicado no Capítulo 3.5.1. Como dito anteriormente, a única diferença entre os algoritmos desenvolvidos está na fase de pré-imputação, portanto, somente a linha 2 sofre alteração ao codificar o GPImpute e o LGPImpute.

Algoritmo 2: Pseudo-código dos algoritmos GPImpute e LGPImpute

Entrada: Base de dados com valores ausentes; Parâmetros do GPImpute
Saída : Base de dados imputada; funções de regressão

```
1  X_va ← base de dados com valores ausentes
2  X_pi ← base de dados pré-imputada
3  para cada atributo A ∈ X_va faça
4      se A tem valores ausentes então
5          Inicializar os indivíduos do GP aleatoriamente;
6          Avaliar os parâmetros da população usando X_pi como base de teste;
7          enquanto g < maxGeracoes ou fitness > 0 faça
8              Aplicar operadores genéticos
9              Avaliar os indivíduos usando X_pi como base de teste
10             f_c ← função de regressão obtida a partir do melhor indivíduo
11         para cada amostra x ∈ A faça
12             se x é ausente então
13                 Aplicar f_c para estimar x em in X_va e X_pi
14 Retornar: Base de dados imputada; funções de regressão.
```

Apêndice B - Tabelas de Análise Estatística

O Método de Holm avalia se os resultados obtidos têm relevância estatística descartando hipóteses falso-positivas. As tabelas a seguir referem-se a este procedimento e indicam quais comparações são válidas considerando cada conjunto de bases estudado em cada algoritmo desenvolvido. As hipóteses descartadas neste método devem ter probabilidade menor que a probabilidade de ocorrência que leva em consideração o número de testes realizados e a proporção de acerto de cada método investigado.

1. Bases com 5% de Valores Ausentes

i	algoritmos	$z = (R_0 - R_i)/SE$	p	Holm	Shaffer
21	GP-MEAN vs. LGP-BEST	3.0929478706587097	0.0019817894378099613	0.004761904761904762	0.004761904761904762
20	KMI vs. GP-MEAN	2.9692299558323616	0.00298547089126885	0.005	0.006666666666666667
19	MC vs. LGP-BEST	2.845512041006013	0.004434008303100672	0.005263157894736842	0.006666666666666667
18	KMI vs. MC	2.721794126179665	0.006492857745083867	0.005555555555555556	0.006666666666666667
17	EC vs. LGP-BEST	2.5980762113533165	0.009374768459434838	0.0058823529411764705	0.006666666666666667
16	EC vs. KMI	2.474358296526968	0.013347575926843111	0.00625	0.006666666666666667
15	GP-BEST vs. GP-MEAN	1.8557687223952262	0.06348653056076746	0.006666666666666667	0.006666666666666667
14	GP-MEAN vs. LGP-MEAN	1.7320508075688779	0.08326451666355028	0.007142857142857143	0.007142857142857143
13	MC vs. GP-BEST	1.6083328927425296	0.10776229039192131	0.007692307692307693	0.007692307692307693
12	MC vs. LGP-MEAN	1.484614977916181	0.13764584551826461	0.008333333333333333	0.008333333333333333
11	EC vs. GP-BEST	1.3608970630898327	0.17354622178785312	0.009090909090909092	0.009090909090909092
10	LGP-BEST vs. LGP-MEAN	1.360897063089832	0.17354622178785334	0.01	0.01
9	EC vs. LGP-MEAN	1.2371791482634844	0.21602058095545867	0.011111111111111112	0.011111111111111112
8	KMI vs. LGP-MEAN	1.2371791482634837	0.21602058095545892	0.0125	0.0125
7	GP-BEST vs. LGP-BEST	1.2371791482634837	0.21602058095545892	0.014285714285714287	0.014285714285714287
6	KMI vs. GP-BEST	1.1134612334371354	0.2655103889538738	0.016666666666666666	0.016666666666666666
5	EC vs. GP-MEAN	0.49487165930539334	0.6206907170753548	0.02	0.02
4	EC vs. MC	0.24743582965269667	0.8045709480174359	0.025	0.025
3	MC vs. GP-MEAN	0.24743582965269667	0.8045709480174359	0.03333333333333333	0.03333333333333333
2	KMI vs. LGP-BEST	0.12371791482634834	0.9015386266571278	0.05	0.05
1	GP-BEST vs. LGP-MEAN	0.12371791482634834	0.9015386266571278	0.1	0.1

Tabela de Holm / Shaffer para $\alpha = 0.10$

O procedimento de Holm rejeita as hipóteses com p-valor ≤ 0.0055.

2. Bases com 10% de Valores Ausentes

i	algoritmos	$z = (R_0 - R_i)/SE$	p	Holm	Shaffer
21	GP-MEAN vs. LGP-BEST	3.464101615137755	5.320055051392219E-4	0.004761904761904762	0.004761904761904762
20	MC vs. LGP-BEST	2.598076211353317	0.009374768459434826	0.005	0.006666666666666667
19	KMI vs. GP-MEAN	2.474358296526968	0.013347575926843111	0.005263157894736842	0.006666666666666667
18	EC vs. LGP-BEST	2.3506403817006194	0.018741136789596654	0.005555555555555556	0.006666666666666667
17	LGP-BEST vs. LGP-MEAN	1.8557687223952253	0.06348653056076756	0.0058823529411764705	0.006666666666666667
16	GP-BEST vs. GP-MEAN	1.7320508075688774	0.08326451666355035	0.00625	0.006666666666666667
15	GP-BEST vs. LGP-BEST	1.732050807568772	0.08326451666355039	0.006666666666666667	0.006666666666666667
14	KMI vs. MC	1.6083328927425298	0.10776229039192128	0.007142857142857143	0.007142857142857143
13	GP-MEAN vs. LGP-MEAN	1.608332892742529	0.10776229039192142	0.007692307692307693	0.007692307692307693
12	EC vs. KMI	1.3608970630898325	0.1735462217878532	0.008333333333333333	0.008333333333333333
11	EC vs. GP-MEAN	1.1134612334371354	0.2655103889538738	0.009090909090909092	0.009090909090909092
10	KMI vs. LGP-BEST	0.9897433186107868	0.32229959587191925	0.01	0.01
9	MC vs. GP-BEST	0.8660254037844395	0.3864762307712322	0.011111111111111112	0.011111111111111112
8	KMI vs. LGP-MEAN	0.8660254037844387	0.3864762307712326	0.0125	0.0125
7	MC vs. GP-MEAN	0.8660254037844379	0.386476230771233	0.014285714285714287	0.014285714285714287
6	MC vs. LGP-MEAN	0.7423074889580912	0.45790105544025433	0.016666666666666666	0.016666666666666666
5	KMI vs. GP-BEST	0.7423074889580904	0.4579010554402549	0.02	0.02
4	EC vs. GP-BEST	0.618589574131742	0.5361867719001739	0.025	0.025
3	EC vs. LGP-MEAN	0.49487165930539373	0.6206907170753545	0.03333333333333333	0.03333333333333333
2	EC vs. MC	0.24743582965269742	0.8045709480174351	0.05	0.05
1	GP-BEST vs. LGP-MEAN	0.12371791482634834	0.9015386266571278	0.1	0.1

Tabela de Holm / Shaffer para $\alpha = 0.10$

O procedimento de Holm rejeita as hipóteses com p-valor ≤ 0.0051.

3. Bases com 30% de Valores Ausentes

i	algoritmos	$z = (R_0 - R_i)/SE$	p	Holm	Shaffer
21	GP-MEAN vs. LGP-BEST	3.587819529964103	3.3345495835016 43E-4	0.004761904761904762	0.004761904761904762
20	KMI vs. GP-MEAN	2.969229955832361	0.0029854708912688535	0.005	0.006666666666666667
19	MC vs. LGP-BEST	2.598076211353316	0.009374768459434853	0.005263157894736842	0.006666666666666667
18	EC vs. LGP-BEST	2.103204552047923	0.03544789255246077	0.005555555555555556	0.006666666666666667
17	GP-MEAN vs. LGP-MEAN	2.1032045520479223	0.0354478925524608	0.0058823529411764705	0.006666666666666667
16	KMI vs. MC	1.9794866372215745	0.04776124267510 3684	0.00625	0.006666666666666667
15	GP-BEST vs. GP-MEAN	1.8557687223952253	0.06348653056076756	0.006666666666666667	0.006666666666666667
14	GP-BEST vs. LGP-BEST	1.7320508075688774	0.08326451666355035	0.007142857142857143	0.007142857142857143
13	EC vs. KMI	1.484614977916181	0.13764584551826461	0.007692307692307693	0.007692307692307693
12	LGP-BEST vs. LGP-MEAN	1.4846149779161806	0.13764584551826475	0.008333333333333333	0.008333333333333333
11	EC vs. GP-MEAN	1.48461497791618	0.1376458455182649	0.009090909090909092	0.009090909090909092
10	KMI vs. GP-BEST	1.1134612334371357	0.26551038895387374	0.01	0.01
9	MC vs. LGP-MEAN	1.1134612334371357	0.26551038895387374	0.011111111111111112	0.011111111111111112
8	MC vs. GP-MEAN	0.9897433186107867	0.32229959587191936	0.0125	0.0125
7	KMI vs. LGP-MEAN	0.8660254037844387	0.3864762307712326	0.014285714285714287	0.014285714285714287
6	MC vs. GP-BEST	0.8660254037844387	0.3864762307712326	0.016666666666666666	0.016666666666666666
5	EC vs. LGP-MEAN	0.6185895741317424	0.5361867719001737	0.02	0.02
4	KMI vs. LGP-BEST	0.6185895741317419	0.536186771900174	0.025	0.025
3	EC vs. MC	0.49487165930539334	0.6206907170753548	0.03333333333333333	0.03333333333333333
2	EC vs. GP-BEST	0.37115374447904537	0.7105230229164893	0.05	0.05
1	GP-BEST vs. LGP-MEAN	0.24743582965269706	0.8045709480174354	0.1	0.1

Tabela de Holm / Shaffer para $\alpha = 0.10$

O procedimento de Holm rejeita as hipóteses com p-valor ≤ 0.0057.

Apêndice C - Artigos Produzidos

A pesquisa realizada neste trabalho culminou na produção de dois artigos científicos. O primeiro aborda o algoritmo GPImpute e os resultados obtidos a partir do mesmo. Este artigo foi submetido para o Simpósio Brasileiro de Pesquisa Operacional - SBPO que ocorrerá em Setembro de 2016 no em Vitória, Espirito Santo.

O segundo artigo produzido é referente ao LGPImpute e os resultados obtidos a partir deste método. Este artigo foi submetido para a quinta Conferência Brasileira de Sistemas Inteligentes, *5th Brazilian Conference on Intelligent System* - BRACIS, que ocorrerá em Outubro de 2016 em Recife, Pernambuco. A seguir encontra-se anexos os artigos produzidos na íntegra.