UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ABORDAGEM DE COMPUTAÇÃO PARALELA PARA GERAÇÃO DE BASES DE REGRAS *FUZZY* EM PROBLEMAS DE GRANDE VOLUME E DE ALTA DIMENSIONALIDADE DOS DADOS USANDO ALGORITMOS GENÉTICOS MULTIOBJETIVO DE ORDENAÇÃO POR NÃO DOMINÂNCIA

MAYKON ROCHA SANTANA

ORIENTADORA: PROFA. DRA. HELOISA DE ARRUDA CAMARGO

São Carlos - SP

Julho/2022

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ABORDAGEM DE COMPUTAÇÃO PARALELA PARA GERAÇÃO DE BASES DE REGRAS *FUZZY* EM PROBLEMAS DE GRANDE VOLUME E DE ALTA DIMENSIONALIDADE DOS DADOS USANDO ALGORITMOS GENÉTICOS MULTIOBJETIVO DE ORDENAÇÃO POR NÃO DOMINÂNCIA

MAYKON ROCHA SANTANA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Orientadora: Profa. Dra. Heloisa de Arruda Camargo

São Carlos - SP

Julho/2022

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Tese de Doutorado do candidato Maykon Rocha Santana, realizada em 28/07/2022.

Comissão Julgadora:

Profa. Dra. Heloisa de Arruda Camargo (UFSCar)

Prof. Dr. Diego Furtado Silva (UFSCar)

Prof. Dr. Murilo Coelho Naldi (UFSCar)

Prof. Dr. Angelo Conrado Loula (UEFS)

Prof. Dr. Graçaliz Pereira Dimuro (FURG)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

A todos aqueles que, de alguma forma, me apoiaram nessa caminhada

AGRADECIMENTO

À minha família por ter me guiado por esse caminho, em especial, a minha irmã Lorena, à minha prima Manuela e aos meus pais, Jário (*in memoriam*) e Hirêne que com garra e esforço me propiciaram toda a educação necessária para que eu chegasse até aqui.

À minha noiva, Marcela Aniceto, que muito me apoiou e torceu por mim todos esses anos nunca me deixando desanimar ante qualquer obstáculo.

À minha orientadora, Profa. Dra. Heloisa de Arruda Camargo, pela dedicação, paciência e por todos os ensinamentos nesses anos.

À Marinete e ao Carlos, pais do Melo e à Dorotí, mãe da Marcela, por me aturarem nos meus dias de folga.

Aos queridos amigos Eduardo Fernando, Natália França, Vinícius Caridá, Tiago Pinho, Elaine Cissa Gatto e Suzane Carol pelos momentos de lazer, pela ajuda nos momentos necessários e por propiciar discussões enriquecedoras nesses anos.

Agradeço também aos colegas Matheus Pires, Marcos Cintra e Adinovam Pimenta que me deram dicas valiosas e o suporte necessário para que o trabalho fosse desenvolvido.

E não posso esquecer do pessoal que tentou me atrapalhar bastante: Jéssica Poellnitz (Jeh), Gustavo Sales Barbosa (Filô, Filord, Filong, Filurso, Filoturno, Filord do Amor, Filô Pão, FilôCaré, ...), Carol Raimundo (Carol), Diogo Melo (Melo), Lorrana Morelli (Lô), Richard Valefuogo (Rich), Marcela Valefuogo, Caio Evaristo (Caião), Jéssica Bianchi, Lucas Zago (Bixão), Bruna Canduzin, Alan Henrique (Baianão), Tamyris Marconi, Wellington Puerta (Punho), André Oliveira, Pablo Matos, Naiana Varges, Thales Sinelli e tantos outros. Na verdade, não ajudaram em nada nesse trabalho, mas o Rich disse que seria injusto não os mencionar já que fizeram meus dias mais felizes durante esses anos.

Agradeço também aos professores Dr. Murilo Coelho Naldi, Dr. Diego Furtado Silva, Dra. Graçaliz Pereira Dimuro e Dr. Angelo Conrado Loula que aceitaram participar dessa banca.

Por fim, agradecimento à CAPES pelo suporte financeiro. "O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001".

Meu muito obrigado a todos.

"Filô, eu vou ser o seu braço direito".

Diogo Melo

RESUMO

O desenvolvimento de Sistemas *Fuzzy* Baseados em Regras (SFBR) traz consigo a necessidade do tratamento de questões relacionadas ao desejável balanceamento entre os objetivos conflitantes de acurácia (medida relacionada à capacidade do sistema de descrever um dado problema) e de complexidade (medida relacionada à interpretabilidade dos SFBR de acordo com o número de regras e de antecedentes das regras). Essa situação também ocorre quando se considera problemas onde os dados de entrada possuem grande volume e alta dimensionalidade. Todavia, nesses casos, surge uma questão: a quantidade de regras é tão grande (milhares ou até mesmo milhões de regras) que a extração de Bases de Regras *Fuzzy* (BRF) acuradas e interpretáveis por meio dos algoritmos sequenciais clássicos torna-se computacionalmente muito custosa e muitas vezes inviável. Nesse contexto, a Computação Paralela surge como um meio para, a partir de uma grande conjunto de regras, possibilitar a extração de BRF que sejam ponderadas entre a acurácia e a complexidade. Sendo assim, nesse trabalho é proposta uma abordagem de computação paralela para geração de BRF a partir de conjuntos de regras com grande volume e alta dimensionalidade. A ideia é que, usando a Computação Paralela e os Sistemas *Fuzzy* Evolutivos Multiobjetivos (SFEMO), seja possível extrair BRF acuradas e compactas (menos complexas | mais interpretáveis) a partir de um grande conjunto de regras. Para a geração dos SFEMO foram usados os Algoritmos Genéticos Multiobjetivo de Ordenação por não Dominância NSGA-DO, MNSGA-DO e o clássico NSGA-II. Cada um desses algoritmos foi testado para geração de BRF a partir de conjuntos de regras *Fuzzy* obtidos por meio do método FCA-*Based*. Com os testes percebeu-se que, com o uso dos AGMO NSGA-DO e MNSGA-DO, foi possível alcançar, além dos dois objetivos principais, uma melhor distribuição das soluções ao longo da Fronteira de Pareto em comparação com as soluções obtidas pelo NSGA-II. Os testes demonstraram que, por meio da abordagem proposta, foi possível a extração de BRF acuradas e compactas a partir de grandes conjuntos de regras. Além disso, testes foram realizados seguindo um método baseado na abordagem estado-da-arte FARC-HD. Foi possível perceber que as BRF obtidas por meio da abordagem aqui proposta possuíam, em geral, maior acurácia em relação às BRF extraídas por intermédio do método baseado na abordagem FARC-HD.

Palavras-chave: Sistemas Baseados em Regras *Fuzzy*; Algoritmos Genéticos Multiobjetivo; Sistemas *Fuzzy*; Computação Paralela; Problemas de grande volume e alta dimensionalidade; Aprendizado de Máquina.

ABSTRACT

The development of Rule-Based Fuzzy Systems (FRBS) brings with it the need to address issues related to the desirable balance between the conflicting objectives of accuracy (measure related to the system's ability to describe a given problem) and complexity (measure related to interpretability of the FRBS according to the number of rules and rule antecedents). This situation also occurs when considering problems where the input data have large volume and high dimensionality. However, in these cases, a question arises: the number of rules is so large (thousands or even millions of rules) that the extraction of accurate and interpretable Fuzzy Rule Bases (FRB) through classical sequential algorithms becomes computationally very costly and often unfeasible. In this context, Parallel Computing appears as a means to, from a large set of rules, enable the extraction of FRB that are weighted between accuracy and complexity. Therefore, this work proposes a parallel computing approach to generate FRB from high-volume and high-dimensional rule sets. The idea is that, using parallel computing and Multiobjective Evolutionary Fuzzy Systems (MOEFS), it is possible to extract accurate and compact (less complex | more interpretable) FRB from a large set of rules. For the generation of MOEFS, the Multiobjective Non-Dominated Sorting Genetic Algorithms NSGA-DO, MNSGA-DO and the classic NSGA-II were used. Each of these algorithms was tested to generate FRB from sets of Fuzzy rules obtained through the FCA-Based method. With the tests it was noticed that, with the use of AGMO NSGA-DO and MNSGA-DO, it was possible to achieve, in addition to the two main objectives, a better distribution of the solutions along the Pareto Frontier compared to the solutions obtained by the NSGA -II. The tests showed that, through the proposed approach, it was possible to extract accurate and compact BRFs from large sets of rules. Furthermore, tests were performed following a method based on the state-of-the-art FARC-HD approach. It was possible to perceive that the BRF obtained through the approach proposed here had, in general, greater accuracy in relation to the BRF extracted through the method based on the FARC-HD approach.

Keywords *– Fuzzy Rule Based Systems; Multiobjective Genetic Algorithms; Fuzzy Systems; Parallel Computing; Large volume and high dimensionality problems; Machine Learning.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AEMO	Algoritmo Evolutivo Multiobjetivo
AG	Algoritmo Genético
AGMO	Algoritmo Genético Multiobjetivo
AWS	*Amazon Web Service*
BC	Base de Conhecimento
BD	Base de Dados
BR	Base de Regras
BRF	Base de Regras *Fuzzy*
CIG	*Computacional Intelligence Group*
DM	Distancia de Multidão (do inglês "*Crowd Distance*")
FBDT	*Fuzzy Binary Decision Trees*
FCA	Análise de Conceito Formal (do inglês "*Formal Concept Analysis*")
FL	Fronteira Limítrofe
FMDT	*Fuzzy Multiway Decision Trees*
GFS	*Google File System*
HDFS	*Hadoop Distributed File System*
IA	Inteligência Artificial
ID	Índice de Distribuição
LAN	*Local Área Network*
MI	Mecanismo de Inferência
MNSGA-DO	*Modified Non-dominated Sorting Genetic Algorithm Distance Oriented*
MOGA	*Multi-Objective Genetic Algorithm*
MPC	*Massive Parallel Computing*
NMP	Número Máximo de partições
NP	Número de Partições
NPGA	*Niched-Pareto Genetic Algorithm*
NRPP	Número de Regras por Partição
NSGA	*Non-dominated Sorting Genetic Algorithm*
NSGA-DO	*Non-dominated Sorting Genetic Algorithm Distance Oriented*
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
NTR	Número Total de Regras

PAES	*Pareto Archived Evolution Strategy*
RDD	*Resillient Distributed Datasets*
SCBRF	Sistema de Classificação Baseado em Regras *Fuzzy*
SFBR	Sistema *Fuzzy* Baseado em Regras
SFEMO	Sistema *Fuzzy* Evolutivo Multiobjetivo
SFG	Sistema *Fuzzy* Genético
SFGMO	Sistema *Fuzzy* Genético Multiobjetivo
SIF	Sistema de Inferência *Fuzzy*
SPEA	*Strength Pareto Evolutionary Algorithm*
SPEA2	*Strength Pareto Evolutionary Algorithm 2*
SQL	*Structured Query Language*
UC	Unidade de Computação
UP	Unidade de Processamento
VEGA	*Vector Evaluated Genetic Algorithm*
WAN	*Wide Area Network*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

Os Algoritmos Evolutivos Multiobjetivos (AEMO) têm sido usados em conjunto com os Sistemas *Fuzzy* e vêm tendo bons resultados quando aplicados na geração dos chamados Sistemas *Fuzzy* Baseados em Regras (SFBR). Alguns desses resultados podem ser encontrados nos trabalhos de Alcalá-Fdez, Alcalá e Herrera (2011), Cárdenas e Camargo (2013), Fazzolari et al. (2013), Lima e Camargo (2014), Cárdenas, Camargo e Tupac (2015), Pimenta e Camargo (2015), Cintra, Camargo e Monard (2016), Ferranti, Marcelloni e Segatori (2016), Ferranti et al. (2017), Tsakiridis, Theocharis e Zaldis (2018), Rodriguez-Mier, Mucientes e Bugarín (2019), Araujo e Krohling (2019), Gou et al. (2020), Geng, Liang e Jiao (2021) e Aghaeipoor, Javidi e Fernanzes (2022). Esses trabalhos demonstram que estudos nessa linha de pesquisa tem obtido grande destaque nos últimos anos.

Dentre os temas abordados nesses trabalhos está o relacionado aos chamados Sistemas *Fuzzy* Evolutivos Multiobjetivos (SFEMO). Esses sistemas surgem da união entre os Sistemas *Fuzzy* e os AEMO e são desenvolvidos para, em geral, lidar com situações que envolvem objetivos conflitantes entre si (*trade-off* entre os objetivos). Nesse caso, os objetivos conflitantes são, comumente, os objetivos relacionados à acurácia (medida relacionada à capacidade dos Sistemas *Fuzzy* de descrever um dado problema) e à complexidade (medida relacionada à interpretabilidade dos SFBR de acordo com o número de regras e de antecedentes das regras).

Nesse contexto, trabalhos como os de Alcalá-Fdez *et al.* (2011), Ferranti *et al.* (2016, 2017), Elkano *et al.* (2020), e Aghaeipoor *et al.* (2022) usam os AEMO para obter Bases de

Regras *Fuzzy* (BRF) geradas com vistas ao desejado equilíbrio entre os objetivos de acurácia e de complexidade. Outros trabalhos desenvolveram aplicações relativas aos SFEMO obtendo bons resultados nas áreas de Mineração de Dados e Aprendizado de Máquina como apontado em Fernández *et al.* (2015).

Dentre essas aplicações, destacam-se atualmente aquelas que envolvem dados com grande volume (grande quantidade de dados) e/ou alta dimensionalidade (dados com vários atributos). Nesses casos, muitas vezes torna-se inviável carregar todos os dados na memória, o que, segundo Ducange, Fazzolari e Marcelloni (2020), impede, por consequência, a execução de algoritmos sequenciais clássicos, incluindo os de mineração de dados e procedimentos de aprendizado de máquina.

No campo dos SFEMO, pesquisas vêm sendo desenvolvidas para lidar com dados com essas características. Esses trabalhos desenvolvidos fazem, em geral, uso da computação paralela como um meio para acelerar o processamento e possibilitar o tratamento de situações que envolvam contextos onde os conjuntos de dados possuem grande volume e/ou grande dimensionalidade.

Abordagens paralelas para tratamento de problemas com essas características vêm ganhando força nos últimos anos, como pode ser percebido nos trabalhos de Rodríguez-Fdez, Mucientes e Bugarín (2016a), Ferranti *et al.* (2017), Tsakiridis, Theocharis e Zalidis(2018) Elkano *et al.* (2018, 2019), Gou *et al.* (2020) e Aghaeipoor, Javidi e Fernandez (2022). Em alguns desses trabalhos, Algoritmos Genéticos Multiobjetivos (AGMO) são usados para auxiliar na busca por soluções que sejam ponderadas entre a acurácia e a interpretabilidade das BRF geradas a partir de um dado conjunto de dados.

Nesse sentido, o foco do presente trabalho está relacionado ao desenvolvimento de uma abordagem que faz uso da computação paralela para geração de BRF em problemas que envolvem grande volume e alta dimensionalidade dos dados empregando os AGMO. O trabalho se insere na área de *Massive Parallel Computing*[1] (MPC) onde os AGMOs serão usados para

[1] O modelo *Massive Parallel Computing* (MPC) ganhou popularidade nos últimos anos e agora é visto como o modelo padrão para processar dados em grande escala. Um sistema MPC consiste de um conjunto de máquinas que podem se comunicar entre si para realização de operações com dados em grande escala (ITALIANO *et al.*, 2019).

buscar, dentre um grande conjunto de regras *Fuzzy*, um ou mais subconjuntos de regras *Fuzzy* para formar uma ou mais BRF relativas aos dados em questão. A busca se dá visando alta acurácia e baixa complexidade (busca por um sistema interpretável) das BRF geradas.

Para o presente trabalho os AGMO usados serão os algoritmos da classe *Non-dominated Sorting Genetic Algorithm*, sendo eles o *Non-dominated Sorting Genetic Algorithm Distance Oriented* (NSGA-DO), o *Modified Non-dominated Sorting Genetic Algorithm Distance Oriented* (MNSGA-DO) e o clássico *Non-dominated Sorting Genetic Algorithm II* (NSGA-II). Mais detalhes sobre esses algoritmos poderão ser vistos no Capítulo 2.

Portanto, a ideia é que, por meio do uso da computação paralela e dos AGMO citados, seja possível realizar o processamento de um grande conjunto de regras *Fuzzy* para gerar BRF balanceadas entre os objetivos de acurácia e de complexidade.

1.2 Motivação

Na área de SFBR, a busca por modelos que possuam alta acurácia e baixa complexidade tem despertado o interesse de muitos pesquisadores nos últimos anos. Trabalhos como os de Alcalá-Fdez, Alcalá e Herrera (2011), Fazzolari *et al.* (2013b), Cárdenas, Camargo e Tupac (2015), Cintra, Camargo e Monard (2016), Ferranti *et al.* (2017), Rodriguez-Mier, Mucientes e Bugarín (2019), Elkano *et al.* (2020) e Aghaeipoor, Javidi e Fernandez (2022), dentre outros, empregaram esforços para lidar com os objetivos de acurácia e complexidade no processo de geração de SBRF. Os AGMO vêm sendo usados nesse contexto para realizar a busca por BRF que estejam devidamente balanceadas entre esses dois objetivos.

No entanto, a otimização por meio dos AGMO fornece soluções que nem sempre são devidamente equilibradas entre os objetivos devido à baixa diversidade das soluções, o que leva ao benefício de apenas um dos objetivos (PIMENTA; CAMARGO, 2015). No trabalho de Pimenta e Camargo (2015) foi então desenvolvido o AGMO *Non-dominated Sorting Genetic Algorithm Distance Oriented* (NSGA-DO) visando possibilitar o aumento da diversidade da população a fim de se obter uma melhor distribuição das soluções na Fronteira de Pareto e uma melhor ponderação entre os objetivos de acurácia e interpretabilidade.

Entretanto, em situações onde os conjuntos de regras usados para obtenção das BRF possuíam grande volume (grande quantidade de dados) e/ou alta dimensionalidade (dados com vários atributos) o uso de abordagens sequenciais tradicionais não possibilitou a geração de BRF. Nessas situações, o uso dos modelos sequenciais clássicos torna-se computacionalmente muito custoso e muitas vezes inviável (milhares e até mesmo milhões de regras a serem processadas).

Assim, surgiu a proposta de desenvolver uma abordagem que faz uso da Computação Paralela para geração de BRF devidamente ponderadas entre a acurácia e a interpretabilidade, a fim de, por meio do AGMO NSGA-DO desenvolvidos em Pimenta e Camargo (2015) e também por meio do seu sucessor, o AGMO MNSGA-DO desenvolvido em Machado *et al.* (2021), realizar o processamento de conjuntos de regras com grande volume e/ou grande dimensionalidade.

Nesse sentido, será apresentada, no tópico a seguir, a hipótese desenvolvida para o presente trabalho.

1.3 Hipótese

Com vistas ao contexto e à motivação para a realização do trabalho, a hipótese é a seguinte:

"No domínio de problemas de grande volume e alta dimensionalidade dos dados, os Sistemas Fuzzy Evolutivos Multiobjetivos podem ser desenvolvidos com o uso da Computação Paralela e com o uso dos Algoritmos Genéticos Multiobjetivo de Ordenação por não Dominância NSGA-DO, MNSGA-DO e *NSGAII a fim de se obter Bases de Regras Fuzzy que sejam devidamente distribuídas na Fronteira de Pareto e que ainda sim sejam soluções equilibradas entre os objetivos de acurácia e de complexidade"*.

1.4 Objetivo

Com base na hipótese, o objetivo desse projeto de doutorado é:

- Desenvolver uma abordagem que faz uso da Computação Paralela para geração de Bases de Regras *Fuzzy* em problemas que envolvem grande volume e alta dimensionalidade dos dados usando os Algoritmos Genéticos Multiobjetivo de Ordenação por não-Dominância NSGA-DO proposto em Pimenta e Camargo (2015) e MNSGA-DO proposto em Machado *et al.* (2021) e NSGAII de Deb *et al.*, (2002).

Como objetivos específicos:

- Desenvolver um Sistema *Fuzzy* e aplicar os algoritmos NSGA-DO, MNSGA-DO e o clássico NSGA-II para geração de BRF a partir de conjuntos de dados com vistas às questões relacionadas à acurácia, complexidade, diversidade e quantidade de soluções finais.
- Desenvolver a abordagem proposta por meio da Computação Paralela para auxiliar o processamento desses grandes conjuntos de dados a fim de se obter BRF devidamente distribuídas na Fronteira de Pareto e que mantenham o equilíbrio entre a acurácia e a complexidade das soluções geradas.
- Testar, comparar e analisar a abordagem desenvolvida de acordo com o AGMO usado e de acordo com um método baseado na abordagem estado-da-arte da literatura FARC-HD desenvolvida em Alcalá-Fdez, Alcalá e Herrera (2011).
- Contribuir para pesquisas de código aberto nessa e em outras áreas que possam ser beneficiar da pesquisa aqui desenvolvida
- Compartilhar de forma pública todo o material desenvolvido nessa pesquisa inclusive códigos, algoritmos e passos para reprodução dos experimentos aqui desenvolvidos.

1.5 Justificativa

Os trabalhos de Fazzolari *et al.* (2013), Deb (2015), Fernández *et al.* (2015) e Jaimes e Coello (2015) apontam que os AEMO podem ser usados para encontrar soluções com um nível adequado tanto de desempenho quanto de custo computacional desde que o problema abordado possua quantidade restrita de dados. Contudo, quando há um grande volume e/ou alta

dimensionalidade dos dados, os algoritmos sequenciais tradicionais não são capazes de processar todo esses dados.

Ducange, Fazzolari e Marcelloni (2020) destacam que, nos últimos anos, uma série de algoritmos de mineração de dados e aprendizado de máquina foram propostos para extração de conhecimento quando se lida com grandes conjuntos de dados. No entanto, é destacado no mesmo trabalho que extrair conhecimento útil e valor não é uma tarefa trivial quando se lida com dados com essas características.

Nesse contexto, Fernández *et al.* (2015) ainda destaca que há queda de desempenho dos sistemas quando do tratamento de problemas que envolvem dados com grande volume e alta dimensionalidade. Isso ocorre devido ao aumento do tempo computacional dispensado ao tratamento dos dados o que afeta o sistema e leva a situações onde as abordagens tradicionais não conseguem processar todos os dados com tempo e custos computacionais viáveis.

Assim, mostram-se necessários esforços para o desenvolvimento de soluções que usem a computação paralela para tratamento dessas situações, em especial, em problemas que envolvam os SFEMO que tratam de dados com grande volume e/ou alta dimensionalidade, como é o caso do presente trabalho.

Em Sudholt (2015) destacam-se os efeitos da paralelização, ressaltando a capacidade da abordagem em proporcionar a redução do custo computacional, ao passo que conduz ao aumento da exploração do conjunto de possíveis soluções para determinados problemas.

Nesse sentido, as aplicações de computação paralela para aperfeiçoamento de SFBR apresentadas nos trabalhos de Fernandez, Rio e Herrera (2016), Rodríguez-Fdez, Mucientes e Bugarín(2016a), Ferranti *et al.* (2017), Fernandez, Almansa e Herrera (2017) e Elkano *et al.* (2018, 2019), Ducange, Fazzolari e Marcelloni (2020) e Aghaeipoor, Javidi e Fernandez (2022) mostram-se promissoras. Nesses trabalhos, são apresentadas soluções que usam a computação paralela para a realização do particionamento dos dados a fim de processa-los separadamente. É também destacado, em alguns desses trabalhos, a busca por BRF ponderadas entre os objetivos de interpretabilidade e acurácia. Nesse caso, os AGMO são usados como ferramenta para encontrar soluções balanceadas entre esses objetivos.

O laboratório CIG da UFSCar desenvolve projetos nessa linha de pesquisa, sendo um deles apresentado no trabalho de Pimenta e Camargo (2015). Nesse trabalho, o AGMO

denominado *Non-Dominated Sorting Genetic Algorithm Distance Oriented* (NSGA-DO) é apresentado como uma modificação do tradicional NSGA-II. A abordagem desenvolvida mostrou-se propícia ao desenvolvimento de projetos relacionados à geração de um conjunto de soluções (no caso, as soluções são Bases de Regras *Fuzzy*) com maior diversidade e com o balanceamento entre a acurácia e interpretabilidade.

Outro trabalho desenvolvido em uma parceria entre o laboratório CIG (*Computational Intelligence Group*) da UFSCar – SP e o laboratório LASIC (Laboratório de Sistemas Inteligentes e Cognitivos) da Universidade Estadual de Feira de Santana – UEFS – BA foi proposto em Machado *et al.* (2021). Nesse trabalho foi apresentado um AGMO denominado *Modified Non-Dominated Sorting Genetic Algorithm Distance Oriented* (MNSGA-DO), que visa ajustar o operador de seleção do NSGA-DO para melhorar sua diversidade em problemas de otimização multiobjetivo.

Desse modo, pretende-se, com o presente trabalho, realizar o desenvolvimento de uma abordagem que faz uso da Computação Paralela para auxiliar o processamento de grandes conjuntos de dados a fim de se obter BRF devidamente distribuídas na Fronteira de Pareto e que sejam equilibradas entre a acurácia e a complexidade das soluções geradas.

1.6 Escopo e Limitações

O presente trabalho faz uso de conjunto de regras previamente gerados por meio do método FCA-*Based* a partir de diversos conjuntos de dados que geram diferentes quantidade de regras. Portanto, a abordagem proposta não gera as regras a partir das bases de dados. Assim, a abordagem proposta seleciona, a partir do conjunto de regras geradas pelo FCA-*Based* um ou mais subconjuntos de regras para formar um ou mais BRF finais.

Em relação aos testes, dois blocos de testes com os conjuntos de regras obtidos por meio do método FCA-*Based* foram formados. O Bloco 1 é formado com conjuntos de regras de até 10.000 regras e no máximo 10 variáveis em cada regra (9 antecedentes e 1 consequente). O Bloco 2 é formado com regras que vão de cerca de 10.000 até cerca de 1 milhão de regras e variáveis que vão de 10 (9 antecedentes e 1 consequente) até 21 variáveis (20 antecedentes e 1 consequente).

A abordagem sequencial é testada e limitada apenas ao Bloco 1. Já a abordagem paralela é aplicada aos dois blocos de teste e os resultados são comparados e analisados.

1.7 Contribuições

Nesta sessão é apresentada uma sumarização das contribuições do presente trabalho. Algumas dessas contribuições foram apresentadas no artigo Santana e Camargo (2019) na *Brazilian Conference on Intelligent Systems* - BRACIS.

- Investigação da aplicação dos Algoritmos Genéticos Multiobjetivo de Ordenação por não Dominância NSGA-DO, MNSGA-DO e NSGA-II no contexto de Computação Paralela para extração de BRF a partir de conjuntos de regras obtidos pelo método FCA-*Based*;
- Investigação do comportamento do AGMO MNSGA-DO no contexto de SFBR;
- Investigação do poder de representação e de compactação das BRF obtidas a partir dos conjuntos de BRF extraídas pelo método FCA-*Based*;
- Contribuição na direção de pesquisas voltadas ao uso da Computação Paralela na área de Sistemas Fuzzy Evolutivos Multiobjetivos;
- Contribuição na direção de um modelo de aplicação dos diversos AGMO presentes no *framework* de código aberto jMetal em ambientes distribuídos;
- Contribuição em relação ao desenvolvimento de um Sistema *Fuzzy* no *framework* de código aberto jMetal, possibilitando que esse possa ser aplicado em ambientes distribuídos;
- Contribuição na direção do desenvolvimento de uma abordagem colaborativa entre o *framework* jMetal, o ambiente de desenvolvimento IntelliJ e a plataforma de serviços para computação em nuvem da Amazon – *Amazon Web Service* – AWS;
- Contribuição na direção de possibilitar o processamento de todo um grande conjunto de regras extraído pelo método FCA-*Based* proporcionando uma maior exploração do espaço de busca e, consequentemente, aumentando a probabilidade de se encontrar BRF mais compactas e acuradas em relação à abordagem baseada no método estado-da-arte FARC-HD que processam apenas uma fração das regras;
- Compartilhamento de forma pública do modelo e dos softwares desenvolvidos, para permitir que o trabalho seja reproduzível e, futuramente, possa ser estendido;

1.8 Organização do Trabalho

No Capítulo 2 será apresentada a fundamentação teórica referente à Lógica *Fuzzy*, aos Algoritmos Genéticos (AG), aos Sistemas *Fuzzy* Genéticos Multiobjetivos e à temas relacionados à Computação Paralela. Esses tópicos são a base para a composição deste trabalho. Históricos, terminologias, características gerais dos sistemas que podem ser formados com essas técnicas, *Framework* da área e os AGMOs NSGA-DO, MNSGA-DO e NSGA-II, a serem usados no presente trabalho, serão os temas abordados nesse capítulo.

No Capítulo 3 serão apresentados os artigos relacionados aos Sistemas *Fuzzy*, aos problemas que envolvem dados com grande volume e/ou alta dimensionalidade, à Computação Paralela e aos Algoritmos Genéticos, em especial aos Algoritmos Genéticos Multiobjetivos. Artigos relacionados à área de geração de Bases de Regras *Fuzzy* e ao uso de Algoritmos Genéticos Multiobjetivos nesse contexto serão alguns dos temas tratados.

O Capítulo 4 será dedicado à apresentação da proposta deste trabalho onde será descrita a principal contribuição da pesquisa e as etapas do desenvolvimento do projeto para a geração de Bases de Regras *Fuzzy* em problemas de grande volume e/ou de alta dimensionalidade dos dados usando técnicas de Computação Paralela e os algoritmos NSGA-DO, MNSGA-DO e NSGA-II.

No Capítulo 5 são apresentados os experimentos realizados e os resultados obtidos e, por fim, no Capítulo 6 são apresentadas as conclusões, as principais contribuições e limitações da proposta desenvolvida destacando as contribuições que o mesmo trará para a área de Aprendizado de Máquina.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

Trabalhos, como os de Ferranti *et al.* (2017), Elkano *et al.* (2018, 2020) e de Ducange, Fzzolari e marcelloni (2020), demonstram que os Sistemas *Fuzzy*, os Algoritmos Genéticos e a Computação Paralela podem atuar em conjunto para a geração de Bases de Regras *Fuzzy* ponderadas entre os objetivos de interpretabilidade e acurácia quando do tratamento de problemas que envolvem dados com grande volume e alta dimensionalidade como é o caso do tema do presente trabalho.

A seguir, serão apresentadas as fundamentações teóricas referentes à temas que estão diretamente relacionadas à proposta do presente trabalho.

2.2 Sistemas *Fuzzy*

Quando se usa a linguagem natural para transmitir algum tipo de informação, dados imprecisos são comumente usados. Expressões como "*hoje está um pouco quente*" ou "*ela é muito alta*" são exemplos de expressões com informações imprecisas usadas cotidianamente. No tratamento de problemas computacionais, essas situações também ocorrem e devem ser tratadas de forma adequada. A Lógica Clássica, no entanto, não é capaz de lidar com as imprecisões de certos problemas devido ao fato de ser baseada apenas em termos de verdadeiro e falso.

A Lógica *Fuzzy*, todavia, suporta o tratamento de valores intermediários que se encontram entre o verdadeiro e o falso da Lógica Clássica. Segundo Nguyen e Walker (2006), a Lógica *Fuzzy* pode ser usada em situações onde haja indefinições do problema, já que a mesma lida com informação incerta, incompleta, imprecisa ou vaga além de tratar de problemas complexos em que a modelagem matemática é difícil de ser implementada.

A Lógica *Fuzzy*, concebida pelo matemático e cientista da computação Lotfi Zadeh, tem sido aplicada com sucesso em diversos seguimentos. Singh *et al.* (2013) citam as áreas de redes de computadores, *soft computing*, roteamento de veículos e sistemas industriais como áreas de destaque.

Da Lógica *Fuzzy* surgem, então, os chamados Sistemas *Fuzzy* que, segundo Castro e Camargo (2004) são sistemas usados para representar e processar informação linguística com mecanismos para lidar com a incerteza e a imprecisão.

No tópico a seguir, será abordado um tipo especial de Sistema *Fuzzy*, chamado de Sistema *Fuzzy* Baseado em Regras.

2.2.1 Sistema *Fuzzy* Baseados em Regras

Cintra e Camargo (2007) caracterizam os Sistemas *Fuzzy* Baseados em Regras – também conhecidos como Sistemas de Inferência *Fuzzy* (SIF) – como sistemas compostos por dois componentes principais chamados de Base de Conhecimento (BC) e Mecanismo de Inferência (MI).

A Base de Conhecimento é formada pela Base de Dados (BD) e pela Base de Regras (BR). A BD contém as definições dos conjuntos *Fuzzy* relacionados aos termos linguísticos usados nas regras *Fuzzy* e a BR armazena o conjunto de regras *Fuzzy* que modelam um dado problema.

Cintra e Camargo (2007) afirmam que o Mecanismo de Inferência é tido como o responsável pela aplicação de um processo de raciocínio. Esse processo usa as inferências para derivar as saídas do sistema de acordo com as regras e fatos conhecidos.

Os tipos mais comuns de Sistemas de Inferência Fuzzy são os modelos do tipo Mamdani (1977) e do tipo Takagi e Sugeno (1985) (SHLEEG; ELLABIB, 2013).

O método Mamdani usa a técnica de defuzificação de uma saída *Fuzzy,* enquanto o método Takagi-Sugeno usa uma média ponderada para calcular a saída *Crisp*[2]. Assim, diz-se que o método Mamdani possui funções de pertinência de saída enquanto o método Takagi-Sugeno não as possui.

Dessa forma, o SIF Mamdani converte o resultado do processo de inferência (conjunto *Fuzzy*) em uma saída numérica exata. Em contrapartida, o SIF Takagi-Sugeno gera na saída um número exato não havendo a necessidade da realização de conversão ao final do processo.

O SIF Mamdani tem seus principais elementos apresentados na Figura 2-1 a seguir.

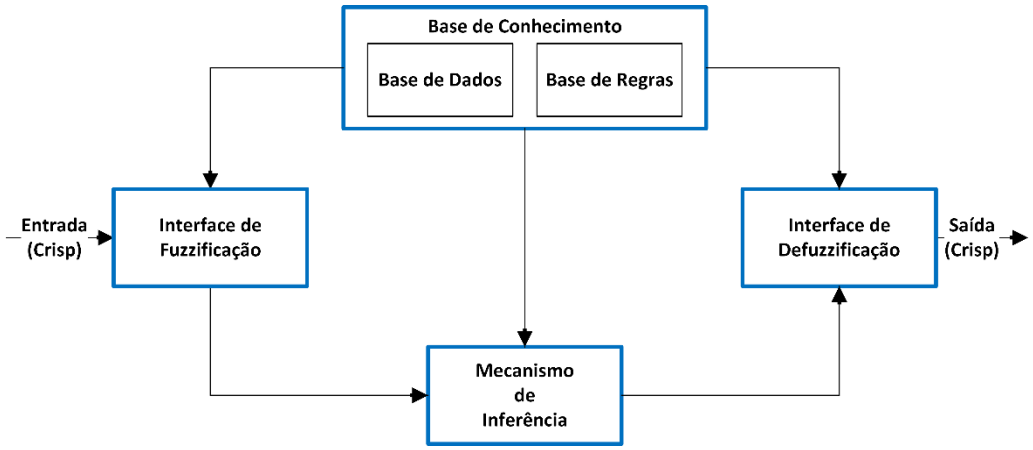

Figura 2-1 - Representação dos elementos de Sistema *Fuzzy* Baseado em Regras Mamdani

Como pode ser visto, o SIF Mamdani possui várias etapas. Na primeira delas, chamada de Interface de Fuzzificação, os dados de entradas são transformados em variáveis *Fuzzy*. A seguir, o Mecanismo de Inferência é executado levando a ocorrência de operações com conjuntos *Fuzzy*. Esse processo usa a Base de Conhecimento, constituída da Base de Dados e da Base de Regras. A BD é responsável por definir as funções de pertinência dos conjuntos *Fuzzy*. A Base de Regras é responsável por armazenar as regras que possuem variáveis linguísticas criadas de acordo com o problema em questão.

A última etapa é chamada de Interface de Defuzzificação. Nessa etapa, são usados os conjuntos *Fuzzy* obtidos no processo de inferência. A Interface de Defuzzificação interpreta os

[2] Na teoria dos conjuntos *Fuzzy*, conjuntos clássicos são frequentemente chamados de conjuntos *Crisp*. As saídas de dados com valores não *Fuzzy* são chamadas de saídas *Crisp*.

conjuntos *Fuzzy* e gera saídas não *Fuzzy* (numéricas) para que as aplicações práticas possam usufruir das respostas obtidas pelos sistemas desenvolvidos.

Os sistemas *Fuzzy* são, portanto, ferramentas importantes para o tratamento de situações onde a informação é imprecisa.

2.2.1.1 Regras e Base de Regras Fuzzy

As regras *Fuzzy* são criadas a partir do uso do operador de implicação *SE* (antecedente) *ENTÃO* (consequente). As Regras *Fuzzy*, portanto, podem ser escritas no formato:

SE (X é A) ***ENTÃO*** (Y é B)

sendo X e Y variáveis linguísticas e A e B subconjuntos *Fuzzy* definidos no universo de discurso de X e Y.

Vale ressaltar que as Regras *Fuzzy* podem assumir formatos de proposições compostas. Nesses casos os operadores lógicos **E** e **OU** são usados gerando regras do tipo:

SE (X1 é A1) **E** (X2 é A2) **ENTÃO** (Y1 é B1) **OU (**Y2 é B2)

onde as variáveis X1, X2, Y1 e Y2 são variáveis linguísticas e A1, A2, B1 e B2 são subconjuntos *Fuzzy* definidos no universo do discurso de X1, X2, Y1 e Y2.

Em um Sistema *Fuzzy*, como o apresentado na Figura 2-1, um dos componentes do Mecanismo de Inferência é a Base de Regras *Fuzzy* que contém um conjunto de Regras *Fuzzy*. A avaliação dessas regras se dá pelas medidas de acurácia e de interpretabilidade. A acurácia é uma medida relacionada à capacidade do conjunto de regras em representar a realidade. A interpretabilidade é uma medida de complexidade e está relacionado com a capacidade de expressar a realidade de uma forma compreensível ao ser humano.

A expectativa, portanto, é que o conjunto de regras *Fuzzy* de um sistema possuam alta acurácia e baixa complexidade, ou seja, espera-se que o conjunto de regras seja acurado sem deixar de ser interpretável.

No entanto, a acurácia e a interpretabilidade são medidas que tendem a ser contraditórias, ou seja, quando melhoramos uma delas, em geral, pioramos a outra. Nesse caso, mostra-se necessário o uso de técnicas que auxiliem na busca por um balanceamento entre essas

duas medidas. Os Algoritmos Genéticos, mais especificamente os Algoritmos Genéticos Multiobjetivos, surgiram como ferramenta auxiliadora na busca por esse balanceamento. A seguir os Algoritmos Genéticos serão apresentados.

2.3 Algoritmos Genéticos

Em situações que envolvem processos de otimização e busca, por vezes, são necessários esforços e recursos significativos para resolução dos problemas. Nesse caso, diversas técnicas podem ser usadas e, dentre elas, pode-se citar o Algoritmo Genético.

Especificamente na área de otimização multiobjetivo voltada para o desenvolvimento dos SFBR, foco do presente trabalho, os AGs vêm sendo empregados em problemas relacionados ao balanceamento entre os objetivos de acurácia e interpretabilidade devido a sua capacidade de busca por soluções ponderadas entre objetivos que se deseja alcançar.

Os AG são procedimentos de busca inerentemente paralelos, probabilísticos, projetados para trabalhar em grandes espaços e que utilizam um conjunto de amostras distribuídas para gerar um novo conjunto de amostras (GOLDBERG; HOLLAND, 1988).

Os AG foram criados com o intuito de imitar os processos observados na teoria da evolução natural das espécies e se mostra, ao longo dos anos, capaz de auxiliar nas resoluções de diversos problemas. Um AG possui a estrutura básica apresenta no **Algoritmo 2-1** adaptado de Santana (2015) apud Rezende (2005).

O algoritmo é iniciado com a inicialização do contador de gerações t em 0 (zero). Em seguida, a população P é iniciada a partir da geração de indivíduos de forma aleatória. A seguir, cada um dos indivíduos da população inicial é avaliado. Posteriormente inicia-se o processo de seleção da população por intermédio da população anteriormente gerada, levando em consideração o *fitness* de cada indivíduo. São aplicados os operadores de cruzamento e mutação (mais detalhes sobre os operadores de cruzamento e mutação serão vistos nos tópicos 2.3.5 e 2.3.6 respectivamente). Em seguida, uma nova avaliação é realizada. O processo é então repetido até que surjam indivíduos que representem as soluções desejada para um dado problema.

Algoritmo 2-1 - Algoritmo Genético

```
1  Início
2      t = 0;
3      Gerar População Inicial P(0);
4      Para Todo Cada indivíduo i da população atual P(0) Faça
5          Avaliar fitness do indivíduo i;
6      Fim Para Todo
7      Enquanto critério de parada não for satisfeito Faça
8          t = t + 1;
9          Selecionar população P(t) a partir de P(t-1);
10         Aplicar operadores de cruzamento sobre P(t);
11         Aplicar operadores de mutação sobre P(t);
12         Avaliar P(t);
13     Fim Enquanto
14 Fim
```

2.3.1 Histórico dos Algoritmos Genéticos

Em 1959 o naturalista Charles Robert Darwin publicou o livro "*On the Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*" (DARWIN, 1859). Neste livro, Darwin apresenta a teoria da Seleção Natural das espécies que foi a base para explicar a adaptação das espécies a determinados ambientes de acordo com as características favoráveis ou desfavoráveis a depender da situação[3].

Em 1866 o monge e botânico austríaco Gergor Johann Mendel publicou o trabalho "*Versuche über Pflanzen-hybriden*" (MENDEL, 1866). Essa pesquisa introduziu as leis da hereditariedade dando origem a área chamada de Genética[4].

[3] Disponível em https://www.biography.com/scientist/charles-darwin, acessado em 02 de abril de 2022.
[4] Disponível em https://www.biography.com/scientist/gregor-mendel, acessado em 02 de abril de 2022.

Em Goldberg (1989) é dito que a junção das pesquisas de Charles Robert Darwin e de Gregor Johann Mendel inspiraram a técnica de Inteligência Artificial chamada de Algoritmo Genético.

Os AG foram introduzidos na década de 1960 pelo físico e matemático John Henry Holland. Segundo Mitchell (1998), o objetivo era usar os mecanismos de adaptação natural em sistemas computacionais. Na década de 1980 surgiu a ideia de combinar os AG com outras técnicas de Inteligência Artificial gerando um intenso campo de pesquisa (KOEHN, 1994).

Nas últimas décadas, os AG vêm sendo usados para resolução de diversos problemas, em especial, na área de Aprendizado de Máquina. Trabalhos como os de Ferranti, Marcelloni e Segatori (2016) e o de Fernandez, Rio e Herrera (2016) fazem uso da técnica de AG em conjunto com os Sistemas *Fuzzy* para resoluções dos problemas de tratamento de dados. O uso dos AG nesse contexto fará parte dos temas abordados no presente trabalho.

2.3.2 Terminologia

Os AG usam diversas analogias relacionadas às características dos seres vivos. Na Tabela 2-1 é apresentada as analogias usadas no presente trabalho.

Tabela 2-1 - Nomenclatura usada para descrever Biologia dos Seres Vivos versus Nomenclatura usada nos AG – Adaptado de Pacheco (1999), Linden (2012) e Santana (2015)

Biologia	Algoritmos Genéticos
Cromossomo	Palavra binária, vetor, *string*
Gene	Característica do problema.
Indivíduo	Representado pelo cromossomo. Solução para um dado problema.
População	Conjunto de Indivíduos.
Geração	Indivíduos de uma população, por meio de recombinação e mutação, geram descendentes.

A função de *fitness* ou aptidão é um outro termo que diz respeito à qualidade de um indivíduo em relação à resolução de determinado problema. Segundo Russell e Norvig (2009) e Linden (2012), a função de *fitness* retorna um índice que representa a avaliação de um indivíduo em relação a um dado problema.

2.3.3 Operador de Seleção

Nos AG existe um operador que seleciona indivíduos que, a partir de suas características, darão origem a uma nova população. Esse operador é chamado de Operador de Seleção.

O Operador de Seleção usa o valor do *fitness* de cada indivíduo para definir os indivíduos que serão os progenitores dos indivíduos da próxima população. Nesse processo, os indivíduos mais aptos, ou seja, de maior valor de *fitness*, tem uma probabilidade maior de serem escolhidos como progenitores dos indivíduos da próxima população.

Os principais métodos usados para seleção são:

Método da Roleta: Desenvolvido por Goldberg (1989), é um método onde cada indivíduo é representado em uma roleta virtual, por uma fatia proporcional ao seu valor de *fitness*. Indivíduos com maior *fitness* ocupam uma fatia maior da roleta ao passo de que, indivíduos com menor *fitness* ocupam fatias menores. Isso faz com que os indivíduos de maior *fitness* tenham maior probabilidade de serem selecionados enquanto os indivíduos de menor *fitness* tenham menor probabilidade de serem selecionados.

Um exemplo, extraído de Santana (2015), pode ser visto na Tabela 2-2.

Tabela 2-2 - Exemplo para o Método da Roleta adaptado de Santana (2015)

Indivíduos	*Fitness*	Fatia da Roleta (°)
Ind1	5	18
Ind2	75	270
Ind3	15	54
Ind4	5	18
TOTAL	**100**	**360**

No exemplo da Tabela 2-2, são apresentados 4 indivíduos que possuem os valores de *fitness* correspondentes. Esses valores somados totalizam o valor 100. A proporção na roleta é calculada e o valor correspondente de cada um dos indivíduos é mostrada na tabela. O valor total da fatia somado resulta em 360º, o seja, uma circunferência completa que representa a roleta.

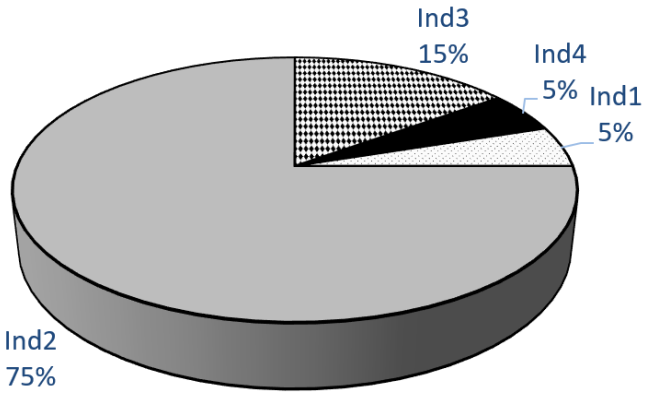

Figura 2-2 - Método da Roleta para os indivíduos da Tabela 2-2 extraído de Santana (2015)

Na Figura 2-2 é apresentada a roleta para cada indivíduo da Tabela 2-2.

Método do Torneio: Nesse método, alguns indivíduos da população são aleatoriamente escolhidos para participar de um torneio. Nesse torneio, dois ou mais indivíduos são comparados e aquele que possuir o maior valor de *fitness* será o escolhido para participar do processo de seleção. Este processo será repetido até que o número de indivíduos que devem ser selecionados para serem os progenitores da próxima população seja atingido.

Um exemplo, extraído de Santana (2015), pode ser visto na Figura 2-3.

Indivíduo	*fitness*
X1	200
X2	100
X3	9500
X4	100
X5	100
X6	10000
X7	1
X8	40

Torneios

X1	X7	X8
X2	X3	X5
X6	X4	X4
X2	X7	X1
X5	X5	X5
X3	X4	X2
X4	X2	X6
X4	X6	X5

Figura 2-3 - Exemplo de seleção de indivíduos pelo Método do Torneio extraído de Santana (2015) apud Linden (2012)

No exemplo da Figura 2-3 os indivíduos sorteados competem no torneio entre 3, sendo o vencedor aquele que tiver o maior valor de *fitness* (no caso, os indivíduos em cinza).

2.3.4 Operador de Elitismo

O Operador de Elitismo foi introduzido pelo cientista da computação Kenneth De Jong em seu artigo "*An Analysis of the Behavior of a Class of Genetic Adaptive Systems*" (JONG, 1975).

Esse operador permite que os melhores indivíduos de uma geração sejam mantidos para a geração seguinte. Sem o uso dessa estratégia, os melhores indivíduos de uma geração podem desaparecer na geração durante as fases de reprodução devido aos processos de cruzamento e mutação. Mitchell (1998) afirma que o elitismo melhora significativamente o desempenho do AG.

Linden (2012) destaca que o gráfico que representa a avaliação do melhor indivíduo no decorrer das gerações de um AG é uma função monotonamente crescente. Nesse caso, a geração posterior terá como seu melhor indivíduo, um indivíduo, no mínimo, como valor de *fitness* igual ao melhor indivíduo da geração anterior.

Em Jong (2006) é dito que o grau de elitismo aumenta a ganancia do algoritmo permitindo o aumento na velocidade de convergência do mesmo. Esse fato, se mal dimensionado, pode ser prejudicial ao AG fazendo com que haja uma convergência indesejada para um ótimo local.

2.3.5 Operador de Cruzamento

O Operador de Cruzamento tem a função de recombinar os cromossomos (repositório das características dos indivíduos) de 2 ou mais indivíduos afim de gerar descendentes.

Em Jong (2006) é dito que em seus estudos, Holland destacava a importância do cruzamento, ou recombinação genética, como fonte de variação da população de indivíduos produzida. Jong (2006) ainda afirma que o cruzamento, juntamente com a mutação (abordada no tópico 2.3.6) fornece o potencial para uma exploração agressiva do espaço de soluções.

Um exemplo, extraído de Santana (2015), pode ser visto na Figura 2-4.

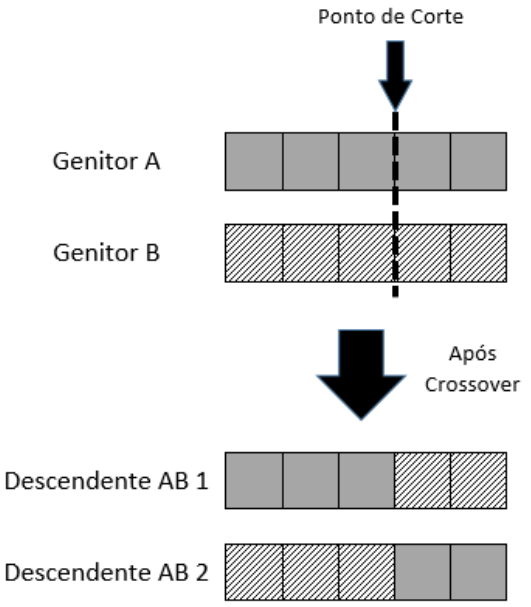

Figura 2-4 - Operador de Cruzamento com 1 ponto de corte – extraído de Santana (2015)

Nesse exemplo, apenas 1 ponto de corte é realizado nos cromossomos dos genitores A e B.

Outro exemplo, com mais de um ponto de corte, pode ser visto na Figura 2-5

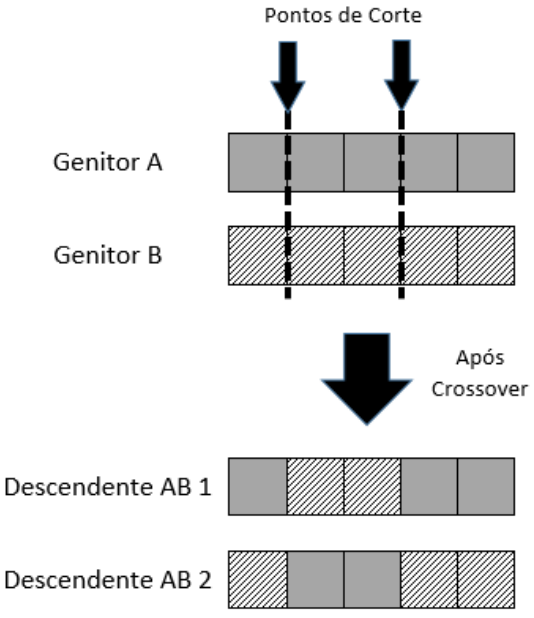

Figura 2-5 - Operador de Cruzamento com 2 pontos de corte – extraído de Santana (2015)

2.3.6 Operador de Mutação

O Operador de Mutação tem a função de modificar alguma característica dos indivíduos por intermédio de alterações aleatórios nos genes dos cromossomos.

Mitchell (1998) e Jong (2006) destacam o Operador de Mutação como o agente provocador de alterações que propicia diversidade em uma população. Essa característica faz com que o AG explore espaços de soluções diferentes e possibilite o encontro de soluções que, em determinadas situações, não seriam possíveis somente com a aplicação dos operadores de Elitismo e Cruzamento.

Um exemplo pode ser visto na Figura 2-6

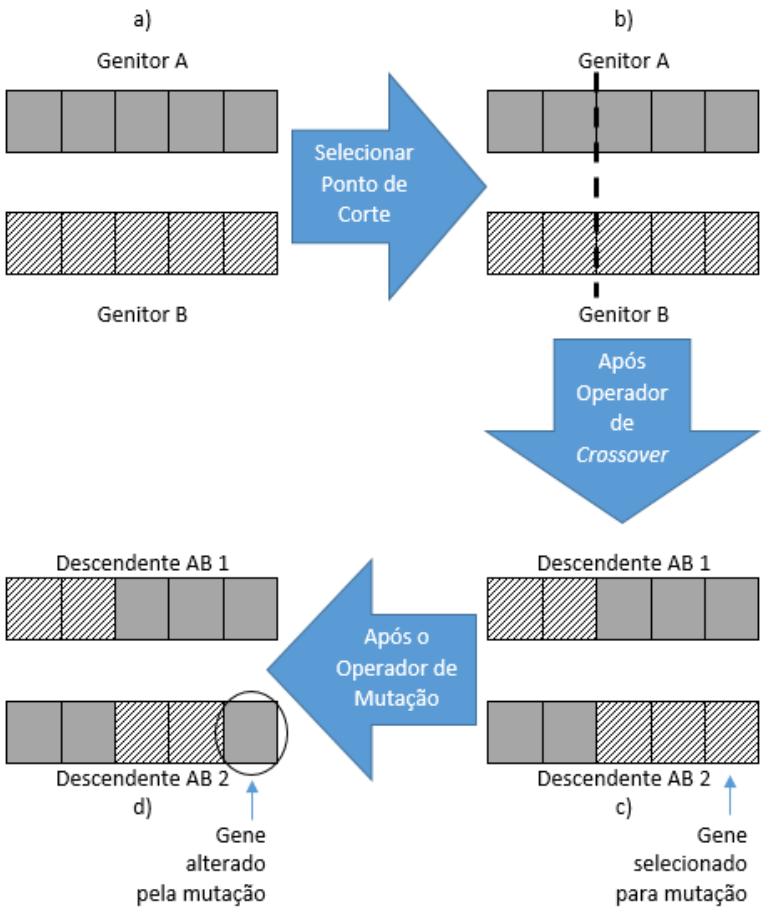

Figura 2-6 - Operador de Cruzamento e Mutação - extraído de Santana (2015) apud Linden (2012)

Para o presente trabalho, os AG serão usados para encontrar um conjunto de regras que comporão a Base de Regras de um Sistema *Fuzzy*.

2.4 Otimização Multiobjetivo

Abraham, Jain e Goldberg (2004) afirmam que é difícil definir todos os aspectos de problemas do mundo real em um único objetivo. É dito que realizar a definição de vários objetivos geralmente oferece uma ideia melhor da tarefa que se deseja realizar. A Otimização Multiobjetivo, portanto, diferentemente da Otimização Mono-objetivo (objetivo único), leva a um conjunto de soluções possíveis de qualidades equivalentes.

Na Otimização Multiobjetivo ocorrem situações em que é preciso lidar com conjunto de objetivos conflitantes. Esses objetivos conflitantes estão relacionados a situações onde, em problemas com dois ou mais objetivos, melhorar a resposta para um deles, pode acarretar em prejuízo de um outro objetivo.

Nesse contexto, usa-se o conceito de Dominância de Pareto. Na Dominância de Pareto o conjunto de soluções para um dado problema multiobjetivo é conhecido como Conjunto de Soluções Não-Dominadas. Esse conjunto está relacionado às soluções que não são piores em nenhum dos objetivos e, pelo menos, melhor em um dos objetivos do que nos outros. É dito, nesse caso, que a solução ideal é aquela que não é dominada por nenhuma outra solução no espaço de busca (ABRAHAM; JAIN; BOLDBERG, 2004). O conjunto de soluções ideais é chamada de Conjunto Pareto-Ótimo.

Matematicamente, para duas soluções x e y em um universo de M funções objetivo com $f_m,\ m = 1, 2, \ldots, M$, é dito que x domina y (expresso por $x \prec y$) se as seguintes condições são satisfeitas (Pimenta (2014)):

- A solução x não é pior que y em todas as funções-objetivo, ou seja, $f_m(x) \leq f_m(y)$
- A solução x é estritamente melhor que a solução y pelo menos em um objetivo, ou seja, $f_m(x) < f_m(y)$ para pelo menos um valor de m.

O conceito de dominância pode ser exemplificado na Figura 2-7.

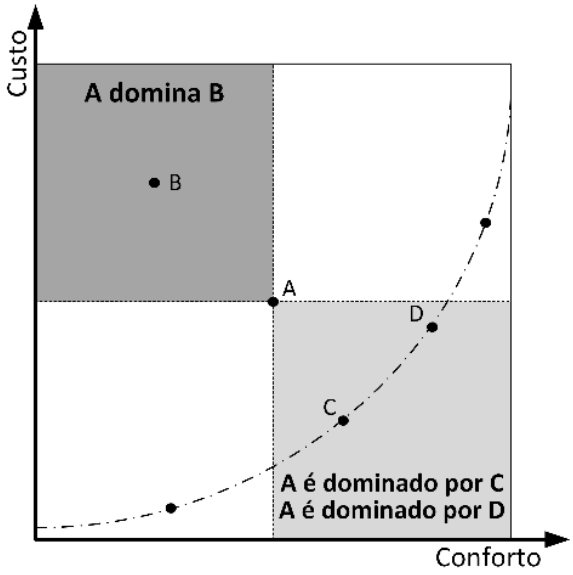

Figura 2-7 – Dominância de Pareto para o problema de Conforto versus Custo (adaptado de Ávila (2006))

A Figura 2-7 representa uma situação em que os objetivos conflitantes Conforto e Custo de um automóvel são mostrados. Na Figura 2-7, o ponto A representa melhor a relação Conforto x Custo em relação ao ponto B (A tem valores maiores de conforto e menores de custo do que B) e, deste modo, é dito que A domina B ($A \prec B$). Entretanto, o ponto C e o ponto D representam melhor a relação Conforto x Custo em relação ao ponto A e, portanto, é dito que tanto o ponto C quanto o Ponto D dominam A e, consequentemente, C e D dominam B.

Para o gráfico da Figura 2-7, a linha que liga os pontos que não são dominados por nenhuma das soluções é chamado de Fronteira de Pareto.

2.4.1 Algoritmo Genético Multiobjetivo

A sugestão de usar algoritmos evolutivos para a resolução de problemas de otimização multiobjetivos surgiu, segundo Abraham, Jain e Goldberg (2004), no fim da década de 1960 na tese de doutorado Rosenberg (1967). Todavia, a primeira implementação real de um Algoritmo Evolutivo Multiobjetivo foi o chamado VEGA ou Algoritmo Genético de Avaliação de Vetores (do inglês "*Vector Evaluated Genetic Algorithm*") apresentado em Schaffer (1984 e 1985).

Posteriormente, o cientista da computação David Edward Goldberg, discípulo do precursor dos AG, Jhon Henry Holland, propôs melhoramentos nos AEMO tentando, por

exemplo, encontrar soluções para problemas relacionados à diversidade da população e à convergência prematura. Foi então que em Fonseca e Fleming (1993) surgiu o hoje conhecido como Algoritmo Genético Multiobjetivo.

Ao longo dos anos, vários algoritmos multiobjetivos foram propostos. Em Coello, Lamont e Veldhuizen (2007) apud Pimenta (2014) são destacadas 2 categorias. A primeira voltada para a simplicidade, onde podem ser citados os algoritmos VEGA, o NSGA (do inglês "*Nondominated Sorting Genetic Algorithm*"), o NPGA (do inglês "*Niched-Pareto Genetic Algorithm*") e o MOGA (do inglês "*Multi-Objective Genetic Algorithm*"). A segunda categoria, voltada para a eficiência onde podem ser citados os algoritmos SPEA (do inglês "*Strength Pareto Evolutionary Algorithm*"), o SPEA2 (do inglês "*Strength Pareto Evolutionary Algorithm II*"), o PAES (do inglês "*Pareto Archived Evolution Strategy*") e o NSGA-II (segunda versão do NSGA).

Outros trabalhos foram sendo realizados com foco no aperfeiçoamento dos AGMO e, dentre eles, destacam-se os trabalhos de Pimenta e Camargo (2015) e Machado *et al.* (2021). Em Pimenta e Camargo (2015) o AGMO chamado NSGA-DO, foi desenvolvido. A sua concepção foi baseada na modificação do conhecido algoritmo genético multiobjetivo NSGA-II. Em Machado *et al.* (2021) o AGMO chamado MNSGA-DO foi apresentado. Ele foi concebido a partir de uma modificação operador de seleção NSGA-DO para melhorar a diversidade das populações.

2.4.1.1 Nondominated Sorting Genetic Algorithms

O primeiro algoritmo genético de ordenação não-dominada chamado de NSGA (do inglês *Non-Dominated Sorting Genetic Algorithm*) foi implementado a partir de uma proposta de Goldberg (1989). A ideia central do NSGA, apresentada em Srinivas e Deb (1994), era classificar os indivíduos em fronteiras não-dominadas e aplicar um método para diversificar o máximo possível as soluções (PIMENTA, 2014).

O NSGA trouxe benefícios no tratamento de problemas multiobjetivos, em especial, nas questões relacionadas à diversidade das soluções. No entanto, problemas relacionados a alta complexidade no procedimento de ordenação de não-dominância e a inexistência do elitismo demonstraram a necessidade da criação de uma versão mais aprimorada do algoritmo.

Foi então que surgiu o NSGA-II, apresentado em Deb *et al.*, (2002). Esse algoritmo foi concebido com a ideia de trazer soluções para os problemas anteriormente citados do NSGA.

2.4.1.2 NSGA-II

O NSGA-II foi proposto com a ideia de classificar indivíduos na chamada fronteira não dominada (com base no conceito de Dominância de Pareto proposto em Pareto (1896)). Nesse algoritmo é definido um procedimento para a seleção de indivíduos a serem adicionados à próxima população, ordenando as soluções com base no critério de Dominância de Pareto. Esse método de seleção pode ser visto na Figura 2-8.

Na Fase A do NSGA-II ocorre a duplicação de cada um dos indivíduos da população fazendo com que a mesma dobre de tamanho e forme um grande grupo contendo as populações A e B que podem ser vistas na Fase B da Figura 2-8. Em seguida, na Fase C, os indivíduos são divididos de acordo com a fronteira de Pareto a qual cada um deles pertence. Esses indivíduos são ordenados seguindo o critério de ordenação por não dominância de Pareto. Pode ser visto no exemplo da Figura 2-8, que nem todos os indivíduos da F3 poderão fazer parte da população final na Fase E. Assim, após a ordenação dos indivíduos por não dominância (e, consequentemente, ordenação pela Fronteira de Pareto), as soluções são copiadas, fronteira por fronteira, para a próxima população. Essa etapa ocorre até que, em algum momento, uma das fronteiras não possa ser integralmente adicionada à próxima população. Isso ocorre devido ao fato de a quantidade de indivíduos da fronteira (no caso, a Fronteira 3) ultrapassar a quantidade máxima de indivíduos que ainda podem ser adicionados ao grupo de indivíduos da próxima população. Essa fronteira pode ser chamada de Fronteira Limítrofe. Nesse caso, o número de indivíduos da Fronteira Limítrofe (no exemplo da Figura 2-8 é a F3) é maior que o espaço ainda disponível para indivíduos serem adicionados à próxima população.

Nesse caso, para definir quais indivíduos, dentre os que compõem a Fronteira Limítrofe, serão adicionados aos últimos espaços ainda disponíveis na próxima população, é utilizado, no NSGA-II, o conceito de Distância da Multidão (DM) (do inglês "*Crowd Distance*"). Nesse caso, o uso da DM tem por objetivo promover, por intermédio do operador de seleção de indivíduos, a diversidade da população ao longo das gerações.

A DM do NSGA-II é então calculada na Fase D para cada uma das soluções da Fronteira Limítrofe (F3) descrita anteriormente. A DM consiste em medir a distância entre dois indivíduos adjacentes a cada indivíduo da população. Nesse caso, os indivíduos mais dispersos (indivíduos com maior distância de multidão) na fronteira têm prioridade no processo de seleção e a população final é então formada. Esse processo é repetido de acordo com o número de gerações estipuladas para o algoritmo em questão.

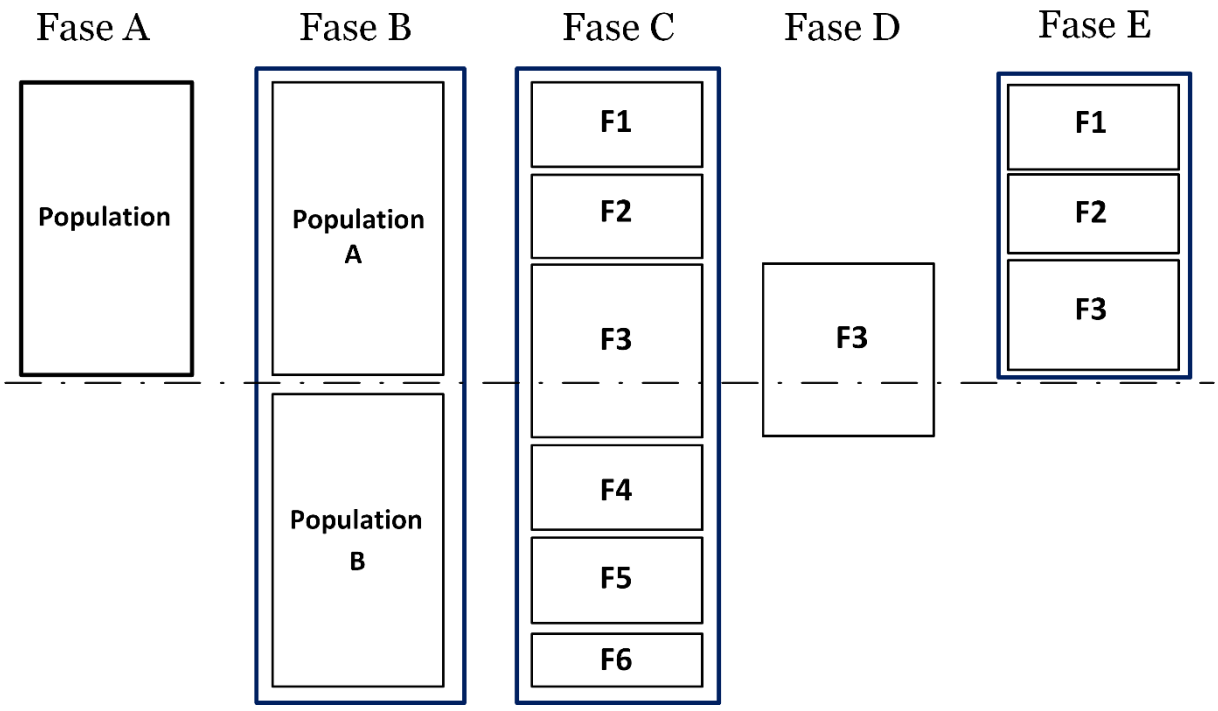

Figura 2-8 - Operações do NSGA-II

2.4.1.3 NSGA-DO

Com a ideia de aperfeiçoar o método de geração das soluções com foco na diversidade das mesmas na fronteira de Pareto, foi apresentada em Pimenta e Camargo (2015) uma nova versão do AGMO NSGA-II chamada de Algoritmo Genético de Ordenação Não-dominada Orientado à Distância ou NSGA-DO (do inglês "*Non-dominated Sorting Genetic Algorithm Distance Oriented*").

No NSGA-DO, a ideia é que as soluções convirjam para os chamados "Pontos Ideais" da Fronteira de Pareto. Os pontos ideais são distribuídos uniformemente ao longo da fronteira de soluções não dominadas, de acordo com o tamanho da fronteira e a quantidade de soluções da fronteira. Nesse caso, em vez de usar o conceito de Distância de Multidão do NSGA-II no

processo de seleção, é usada a distância de cada solução aos Pontos Ideais. Na Figura 2-9, o impacto da diferença entre os critérios de seleção utilizados no NSGA-II e no NSGA-DO pode ser exemplificado.

Na Figura 2-9, F_1 representa a Fronteira de Pareto cujas soluções não são dominadas e F_2 representa uma Fronteira de Pareto onde as soluções são dominadas pelos indivíduos da fronteira F_1. Os círculos cinzas (F_1S_1, F_1S_2, F_1S_3, F_1S_4, F_1S_5, F_2S_1, F_2S_2, F_2S_3 e F_2S_4) representam as soluções encontradas por um AGMO, os pontos pretos (I_1, I_2, I_3, I_4 e I_5) na fronteira F_1 representam os Pontos Ideais e $f1$ e $f2$ são os objetivos conflitantes a serem otimizados.

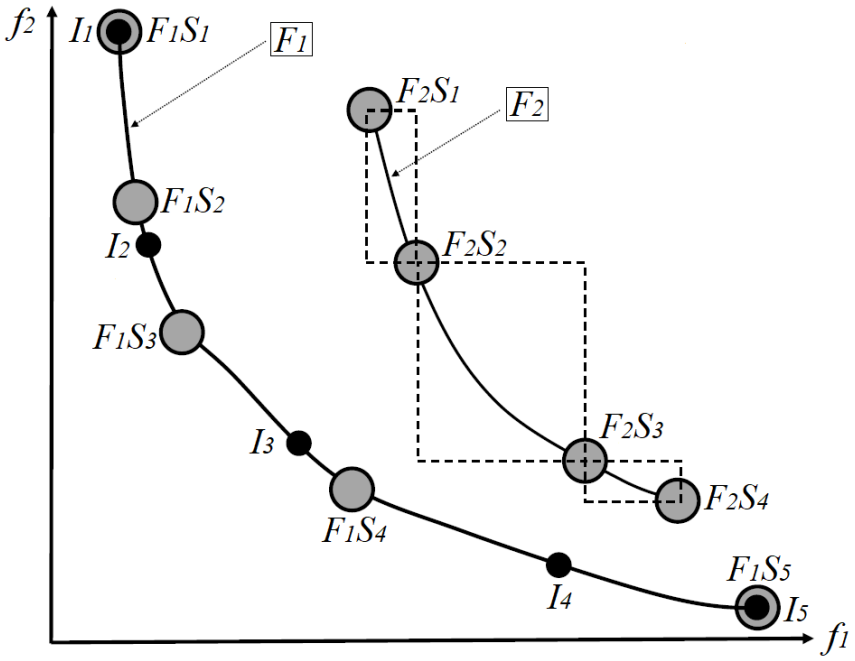

Figura 2-9 - Escolha de soluções por proximidade dos pontos ideais (adaptado de Pimenta e Camargo (2015))

Neste exemplo, considerando que 8 (oito) soluções devem ser selecionadas para a próxima geração, percebe-se que, dependendo do algoritmo aplicado (NSGA-II ou NSGA-DO), os conjuntos de soluções selecionadas serão diferentes.

Com o NSGA-II, as 5 (cinco) soluções da fronteira F_1 (F_1S_1, F_1S_2, F_1S_3, F_1S_4 e F_1S_5) seriam selecionadas e as 3 (três) restantes seriam selecionadas na frente F_2. A DM também é calculada para cada uma das soluções em F_2. As soluções F_2S_1 e F_2S_4 são selecionadas porque estão nas extremidades e, portanto, têm DM tendendo ao infinito. Entre as soluções F_2S_2 e F_2S_3,

a solução F_2S_2 é selecionada porque possui DM maior que o F_2S_3. Portanto, o conjunto final S de soluções obtidas através do NSGA-II é:

$S_{NSGA\text{-}II} = \{F_1S_1, F_1S_2, F_1S_3, F_1S_4, F_1S_5, \mathbf{F_2S_1}, F_2S_2, F_2S_4\}$

Com o uso do NSGA-DO, assim como no NSGA-II, seriam selecionadas as 5 (cinco) soluções da fronteira F_1. As 3 (três) soluções restantes (para completar as 8 (oito) estipuladas) serão selecionados da fronteira F_2. Em F_2, as soluções selecionadas seriam F_2S_2, F_2S_3 e F_2S_4 porque estão mais próximas dos Pontos Ideais do que a solução F_2S_1. Nesse caso, o conjunto final S de soluções obtidas através do NSGA-DO é:

$S_{NSGA\text{-}DO} = \{F_1S_1, F_1S_2, F_1S_3, F_1S_4, F_1S_5, F_2S_2, \mathbf{F_2S_3}, F_2S_4\}$

No caso do NSGA-II, a solução F_2S_3 é descartada mesmo que seja a solução mais próxima de um dos pontos ideais (I_4). Esse fato pode impactar diretamente a convergência do algoritmo, pois quando são selecionadas soluções mais próximas dos Pontos de Ideais, as chances de que, nas próximas gerações, os indivíduos estejam ainda mais próximos desses pontos serão maiores. Portanto, o NSGA-DO seleciona, a cada iteração, soluções mais próximas aos Pontos Ideais, favorecendo a dispersão dessas soluções na fronteira e, consequentemente, aumentando a diversidade da população.

Outro ponto importante a se destacar é que o NSGA-DO usa o processo chamado de seleção por torneio de distância. Nesse processo, um subconjunto de soluções é escolhido para um torneio onde as soluções que possuírem menor distância para o ponto ideal são escolhidas.

O Algoritmo NSGA-DO possui a estrutura básica apresenta no Algoritmo 2-2.

No passo 15 do Algoritmo 2-2, enquanto o resultado da soma do número de soluções armazenadas na nova população com o número de soluções da fronteira atual ($'F_i'$ – que está sendo analisando no momento) for menor ou igual que o tamanho máximo reservado para a população ($'T'$), são copiadas todas as soluções da fronteira atual ($'F_i'$) para P_{n+1}. Essa cópia ocorre até que, ao chegar em determinada fronteira $'F_i'$, percebe-se que todas as soluções dessa fronteira (fronteira atual $'F_i'$) não podem ser adicionadas por completo à nova população $'P_{n+1}'$. Nesse caso o laço "*enquanto*" se encerra.

A partir daí são calculadas as distancias para os Pontos de Ideais das soluções da última fronteira que não pôde ser integralmente adicionada a nova população $'P_{n+1}'$. Ordena-se a

fronteira atual $'F_i'$ de forma decrescente em relação aos Pontos de Ideais. Em seguida é realizada a cópia para P_{n+1} de um subconjunto das soluções ordenadas da fronteira atual ($'F_i'$ – que está sendo analisando no momento). Esse subconjunto terá o tamanho igual ao número de soluções faltantes para que o tamanho da população P_{n+1} seja exatamente igual a T.

Como dito anteriormente, essa abordagem favorece a dispersão das soluções na fronteira e, com isso, propicia o aumento da diversidade na população.

Algoritmo 2-2 - NSGA-DO

Início
    P: População pai;
    Q: População filha;
    T: Tamanho fixo para P e Q;
    F_i: Conjunto de soluções na fronteira j;
    n: Número da geração atual;
    N: Número máximo de gerações;
    Gerar a população inicial P_0 aleatoriamente;
    Criar Q_0 = {};
    Atribuir $n = 0$;
    Realizar Seleção, Cruzamento e Mutação em P_n para gerar a população filha Q_n;
    Fazer $R_n = P_n \cup Q_n$;
    Realizar ordenação por não dominância em R_n, gerando F_i, i = {1, ..., v} em R_n;
    Criar P_{n+1} = {};
    Enquanto $|P_{n+1}| + |F_i| \leq T$ **Faça**
        Copiar as soluções de F_i em P_{n+1};
        $i = i + 1$;
    Fim Enquanto
    Para cada solução em F_i calcular a distância para os pontos ideais;
    Ordenar F_i decrescentemente de acordo a distância para os pontos ideais;
    Copiar as primeiras $T - |P_{n+1}|$ soluções ordenadas de F_i para P_{n+1}
    Se $n \leq N$ **Então**
        Pare;
    Senão
        Atribuir $n = n + 1$ e voltar ao passo 11;
    Fim Se
Fim

Devido a esses fatores, o NSGA-DO foi um dos algoritmos escolhidos para ser usado neste trabalho.

2.4.1.4 MNSGA-DO

Com a ideia de aperfeiçoar ainda mais o método de geração das soluções com foco na diversidade das mesmas na Fronteira de Pareto, foi apresentada em Machado *et al.* (2021) uma nova versão do AGMO NSGA-DO chamada de Algoritmo Genético de Ordenação Não-dominada Orientado à Distância Modificado ou MNSGA-DO (do inglês "*Modified Non-dominated Sorting Genetic Algorithm Distance Oriented*").

A ideia do MNSGA-DO está no fato da mudança em relação tanto na quantidade de Pontos Ideais criados em relação ao NSGA-DO (o número de Pontos ideias é igual a 2 vezes o número de soluções) e também em relação a forma como as soluções são associadas a esses Pontos Ideais. No NSGA-DO, várias soluções podem ser aglutinadas em um mesmo Ponto Ideal e no MNSGA-DO evita-se essa ocorrência, o que corrobora com o espalhamento das soluções ao longo da fronteira.

Assim, durante a execução do MNSGA-DO, se em um dado momento uma determinada solução tende a se associar a um certo Ponto Ideal, primeiramente é verificado se, naquele Ponto Ideal já há uma solução associada a ele. Se sim, busca-se um próximo Ponto Ideal para associar uma dada solução. Esse fato corrobora para que as soluções fiquem melhor distribuídas ao longo da fronteira de Pareto.

2.5 Sistemas *Fuzzy* Genéticos

Alguns trabalhos abordam o uso de AG para o desenvolvimento de Sistemas *Fuzzy*. Os sistemas desenvolvidos com essa estratégia recebem o nome de Sistemas *Fuzzy* Genéticos (SFG). No presente trabalho os AG são usados para gerar um Sistema *Fuzzy* Baseado em Regras. Nesse caso, o AG é usado para desenvolver os componentes do SFBR. Em Castro e Camargo (2004) é dito que os AG são ferramentas usadas para a geração de BRF, para a otimização das BRF, para a geração de funções de pertinência e para a sintonia das funções de pertinência.

Uma categoria particular de SFG são os Sistemas *Fuzzy* Genéticos Multiobjetivos (SFGMO). Para o presente trabalho pretende-se, por meio de um AGMO, gerar SFBR com

equilíbrio entres os objetivos conflitantes acurácia e interpretabilidade. O uso de AGMO nesse contexto, ou seja, na geração de Sistemas *Fuzzy*, forma o chamado SFGMO.

2.5.1 Sistemas *Fuzzy* Genéticos Multiobjetivos

Os SFGMO são usados para encontrar soluções ponderadas entre 2 ou mais objetivos. Em Sistemas *Fuzzy* os objetivos mais explorados são os de acurácia (medida relacionada à capacidade dos Sistemas Fuzzy de descrever um dado problema) e o de complexidade (medida relacionada à interpretabilidade dos SFBR de acordo com o número de regras e de antecedentes das regras). A complexidade, por vezes, descrita também como interpretabilidade tem relação com a capacidade do sistema em expressar um comportamento que seja compreensível ao ser humano.

A acurácia e a complexidade são objetivos que possuem um *trade-off* entre si, ou seja, quando se melhora um, há uma tendência natural para que o outro seja piorado. Em Pimenta (2014) é dito que frequentemente há uma prevalência de um objetivo sobre o outro sendo esse uma dificuldade frequente encontrada em SFBR. Esse fato corroborou com a busca por projeto de SFBR que visam o devido balanceamento entre a acurácia e a complexidade dos SF gerados.

Contudo, encontrar o balanceamento entre a acurácia e a complexidade de um SF é uma tarefa complexa justamente por esses objetivos possuírem um *trade-off* entre si.

Uma forma de se buscar por soluções com essas características é por meio do Aprendizado por Seleção de Regras que é o tema do presente trabalho. Assim como nos trabalhos de Alcalá-Fdez, Alcalá e Herrera (2011), Ferranti *et al.* (2017), Elkano *et al.* (2020), dentre outros, pretende-se usar um AGMO para realizar a seleção de regras *Fuzzy* para geração de BRF com alta acurácia e que sejam também compactas (baixa complexidade).

2.6 Big Data

Durante as últimas décadas, os avanços tecnológicos tornaram o termo *Big Data* amplamente popular nos domínios industrial e acadêmico (SILVA; DIYAN; HAN, 2019). Entretanto, os pesquisadores propõem variadas definições para o termo *Big Data*. Laney

(2001) caracterizou o *Big Data* como a área cujos dados possuem "3 Vs". Esses "Vs" estão relacionados ao Volume (Quantidade de dados a serem processados), Variedade (Dados estruturados, semiestruturados e não estruturados) e Velocidade (Velocidade na qual os dados são gerados, produzidos, criados ou atualizados) dos dados

Ao longo dos anos outros autores como Chen, Mao e Liu (2014), Khan, Uddin e Gupta (2014), Khan *et al.* (2018) e Sun (2018) adicionaram outros "V's" à caracterização do *Big Data* tornando ainda mais difusa a definição do termo. Os outros "V's" são a Variabilidade (número de inconsistências nos dados), o Valor (benefícios que as informações podem retornar), a Veracidade (confiança nos dados), a Validade (precisão e correção dos dados para o uso pretendido), a Vulnerabilidade (segurança dos dados), a Volatilidade (tempo em que os dados estão disponíveis) e a Visualização (formas de representar os dados).

Como o termo carece de uma definição precisa, no presente trabalho optou-se por definir explicitamente que serão tratados os problemas onde os dados possuem grande volume e/ou alta dimensionalidade.

2.7 Grande Volume Alta Dimensionalidade dos Dados

No contexto de aprendizado de máquina, Zhou *et al.* (2017) afirma que a grande quantidade de dados despertou amplos interesses e proporcionou desafios na obtenção de novas ideias sobre vários aplicativos de negócios e comportamentos humanos.

Todavia, Fernández *et al.* (2015) afirmam que há queda de desempenho dos sistemas quando se trata de problemas que envolvem muitos dados e, neste caso, há um aumento exponencial no tempo computacional dispendido para tratamento desses dados, em especial, em problemas que envolvam SFEMO.

A computação paralela surge então como uma forma de lidar com problemas que envolvem dados com grande volume e/ou alta dimensionalidade. Abordagens para lidar com dados com essas características surgiram ao longo dos anos a algumas delas são apresentadas a seguir.

2.7.1 Apache Hadoop

O *Apache Hadoop* é um *framework* projetado para computação escalável e processamento distribuído em *Big Data*. O *Hadoop* é aplicado em problemas de *Big Data*, dividindo os dados de modo a processá-los paralelamente. No fim, os resultados dos processos paralelos são unidos fornecendo uma determinada solução[5].

O *Hadoop* é desenvolvido, em grande parte, na linguagem de programação Java e teve início depois da publicação do artigo referente ao *Google File System* (GFS) (GHEMAWAT; GOBIOFF; LEUNG (2003).

Dentre os componentes do *framework Apache Hadoop* são destacados em Landset *et al.* (2015) e Vohra e Vohra (2016a; b; c) os seguintes módulos:

- *Hadoop Common*
  - Bibliotecas e arquivos usados pelos módulos do *Hadoop*.

- *Hadoop Distributed File System* (HDFS)
  - Sistema de arquivos distribuído do *Hadoop* com suporte à tolerância a falhas.

- *Hadoop Yarn*
  - Plataforma de gerenciamento e agendamento de recursos computacionais em *clusters*.

- *Hadoop MapReduce*
  - Modelo de programação para processamento em larga escala do *Hadoop*. O *MapReduce* é um modelo de programação para processar e gerar grandes conjuntos de dados (DEAN; GHEMAWAT, 2008).

- *Hive*

[5] Disponível em: http://hadoop.apache.org/, acessado em 04 de abril de 2022.

- Software de para banco de dados desenvolvido para consultas e análise dos dados no *Hadoop*. As consultas são realizadas usando a linguagem HIVEQL (semelhante à Linguagem de Consulta Estruturada SQL (do inglês "*Structured Query Language*"))

- *Sqoop*
  - Aplicativo de interface de linha de comandos para transferir dados entre Bancos de Dados relacionais e o *Hadoop*

- *HBase*
  - Banco de Dados *NoSQL* (não relacional) escalável e distribuído projetado para o Apache *Hadoop*.

- *Mahout*
  - Biblioteca de Aprendizado de Máquina de código aberto voltada para a realização de classificações e agrupamentos.

Dentre esses, destacam-se 2 componentes do chamado Ecossistema *Hadoop* (Família de projetos relacionados do *Hadoop*). Esses sistemas são o HDFS e o *Hadoop Map Reduce*[6]. Esses 2 componentes são partes importantes da arquitetura do *framework* e fornecem a estrutura para a manipulação dos dados (HDFS) e para o advento do paralelismo no processo de tratamento dos dados (*Hadoop MapReduce*).

2.7.2 *Apache Spark*

O *Apache Spark*[7] é uma plataforma de código aberto popular para processamento de dados em larga escala, adequada para tarefas de aprendizado de máquina iterativas (MEG *et al.*, 2016). Foi inicialmente apresentado em Zaharia *et al.* (2010) com foco na classe de

[6] O MapReduce é baseado no uso de duas funções denominadas *Map* e *Reduce*. A função *Map* processa um par de chave/valor para gerar um conjunto de respostas intermediário. A função *Reduce* que mescla todos os valores intermediários para gerar o resultado final (DEAN; GHEMAWAT, (2008).

[7] Disponível em: http:// https://spark.apache.org/, acessado em 04 de abril de 2022.

aplicativos que reutilizam um conjunto de dados de trabalho em várias operações paralelas incluindo algoritmos de aprendizado interativo como ferramentas interativas de análise de dados. Em Zaharia *et al.* (2010) e posteriormente em Zaharia *et al.* (2012) foram apresentados os *Resilient Distributed Datasets* (RDD) como uma abstração de memória distribuída que permite aos programadores realizar cálculos na memória em grandes clusters de maneira tolerante àfalhas (característica essa advinda do *Apache Hadoop*). O Spark, segundo Zaharia *et al.* (2012), é eficiente em cálculos iterativos e, portanto, adequado para o desenvolvimento de aplicativos de aprendizado de máquina em larga escala.

Na Figura 2-10 é mostrado o funcionamento simplificado do *Hadoop* e do *Spark*. O *Hadoop* concentra a execução das operações em disco e o *Spark* concentra a executa das operações em memória.

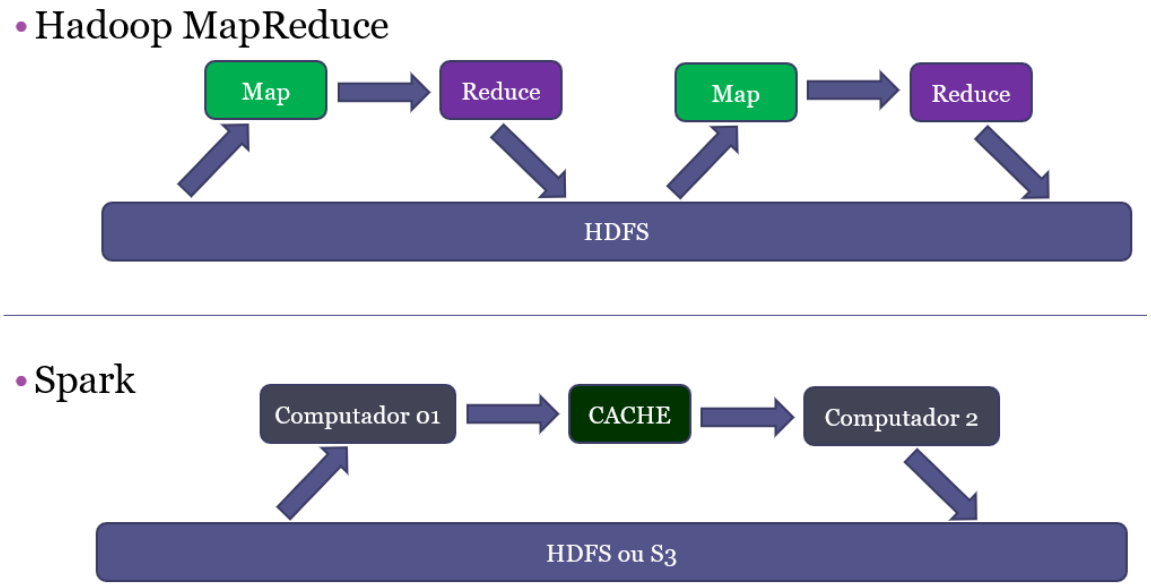

Figura 2-10 - Funcionamento simplificado do *Apache Hadoop* e do *Apache Spark*

2.7.2.1 Resilient Distributed Datasets (RDD)

O *Apache Spark* possui a característica de realizar processamento em memória de trabalho ao passo em que usa os Conjuntos de Dados Distribuídos Resilientes (do inglês "*Resillient Distributed Datasets*" – RDD) e vários modelos de armazenamento de dados para tolerância a falhas, o que minimiza a entrada e saída da rede. Os RDDs são, segundo Zaharia *et al.* (2012), uma abstração de memória distribuída que permite que os programadores realizem cálculos na memória em *clusters* de maneira tolerante a falhas.

Os RDD são, portanto, uma estrutura de dados fundamental do *Spark* permitindo que a computação ocorra de forma distribuída em diferentes nos de um determinado cluster. Cada conjunto de dados no Spark RDD é particionado para que seja possível realizar o processamento em vários nós do cluster.

Como o próprio nome já diz, o RDD é um Conjuntos de Dados Distribuídos Resilientes. A expressão **"Conjunto de Dados"** diz respeito aos dados de trabalho que serão particionados e processados. É dito **"Distribuído"** pelo fato desses mesmos dados serem particionados e processados, de forma distribuída, em diferentes nós de um cluster. E, por fim, é dito **"Resiliente"** por ser tolerante a falhas, sendo capaz de, segundo Zaharia *et al.* (2012), se recuperar de falhas provocadas por partições ausentes ou danificadas.

2.7.2.2 Componentes do Apache Spark

Dentre os componentes do *framework Apache Spark*, são destacados em Meng *et al.* (2016) os seguintes módulos:

- *Core*
  - Base do projeto que fornece funcionalidades distribuídas para o despacho de tarefas, planejamento e entradas e saídas.
- *Spark SQL*
  - Fornece uma linguagem para manipular os dados em linguagens como *Scala*, *Java* ou *Python*.
- *Spark Streaming*
  - Usa o *Spark Core* para executar análises de *streaming* de dados.
- *MLlib*
  - Estrutura de aprendizado de máquina que executa sobre o *Spark Core*.
- *GraphX*
  - Estrutura de processamento gráfico distribuído.

O *Apache Spark* também fornece APIs nas linguagens *Java, Scala*, *Python* e *R* e pode funcionar em conjunto com o *Hadoop* e seus módulos.

Em Carey (2017) é realizada uma comparação entre o *Apache Hadoop* e *Apache Spark* e algumas características desses *frameworks* são apresentadas. Alguns desses pontos são:

- *Redundância*
    - É um dos principais benefícios do uso do *Hadoop* que, por ser uma plataforma distribuída, tem menor propensão à falhas, permitindo que dados subjacentes estejam sempre disponíveis. O *Spark* implementa o RDD com estratégia voltada para tolerância a falhas visando o aumento da confiabilidade.
- *Custo*
    - O *Hadoop* e *Spark* são projetos da *Apache Software Foundation* e, portanto, são softwares livres e *open sources*. O custo da implantação varia com a forma com que se deseja implementá-lo. O tempo gasto, os recursos relacionados à implementação, as habilidades necessárias e o hardware envolvido também impactam nos custos.
- *Velocidade*
    - É relatado pela *Apache Software Foundation*[8] que o *Spark* é executado até 100 vezes mais rápido que o *Hadoop MapReduce*. Isso ocorre porque o *Spark* trabalha na memória, em vez de ler e gravar em, e para, discos rígidos. O *Hadoop MapReduce* lê os dados do *cluster*, executa uma operação e escreve os resultados de volta no *cluster*, o que acarreta um amento no tempo de execução das operações. O *Spark* realiza essas tarefas em um só lugar e próximo ao tempo real, tornando o desempenho várias vezes mais rápido.
- *Generalidade*
    - O *Spark* pode carregar dados de várias fontes de dados, dando muita flexibilidade às equipes. O *Spark* usa como fonte de dados o *Amazon S3*, HDFS ou o *Couchbase*. O *Hadoop* usa o HDFS como fonte de dados.
- *Habilidades*
    - O *Spark* é geralmente destinado a analistas de dados e especialistas e pode ser aplicado a conjuntos de dados profundamente complexos e em constante mudança.

[8] Disponível em: http:// https://spark.apache.org/, acessado em 20 de maio de 2022.

2.8 Computação Paralela

A Computação Paralela é uma forma de computação onde vários cálculos são realizados simultaneamente (ALMASI; GOTTLIEB, 1994). Ela trabalha sobre a ideia de que os problemas podem ser particionados e ter suas partes processadas de forma paralela, ou seja, enquanto uma parte é processada em uma Unidade de Processamento (UP), outra parte, no mesmo momento, também é processada por outra UP.

A Computação Paralela é, em geral, aplicada a problemas onde é necessário um grande poder de processamento e, em alguns casos, grandes quantidades de dados devem ser processadas.

A Computação Paralela pode também ocorrer em um ambiente de Computação Distribuída. Um sistema de Computação Distribuída, segundo Tanenbaum e Steen (2007), é conceituado como uma coleção de computadores independentes entre si que se apresenta ao usuário como um sistema único e coerente.

Tanenbaum e Steen (2007) afirmam que a Computação Distribuída consiste em somar o poder computacional de computadores que estão interligados entre si por meio de uma rede de comunicação. Esta rede pode ser uma rede local (LAN – *Local Area Network*) ou uma rede de longa distância (WAN – *Wide Area Network*). Considerando esses computadores como um sistema único, entende-se que, ao realizar processamento de forma distribuída, a computação se dá de forma paralela uma vez que os processos computacionais são executados ao mesmo tempo, em máquinas e *cores* diferentes. Na Figura 2-11 é mostrado um exemplo desse tipo de arquitetura computacional.

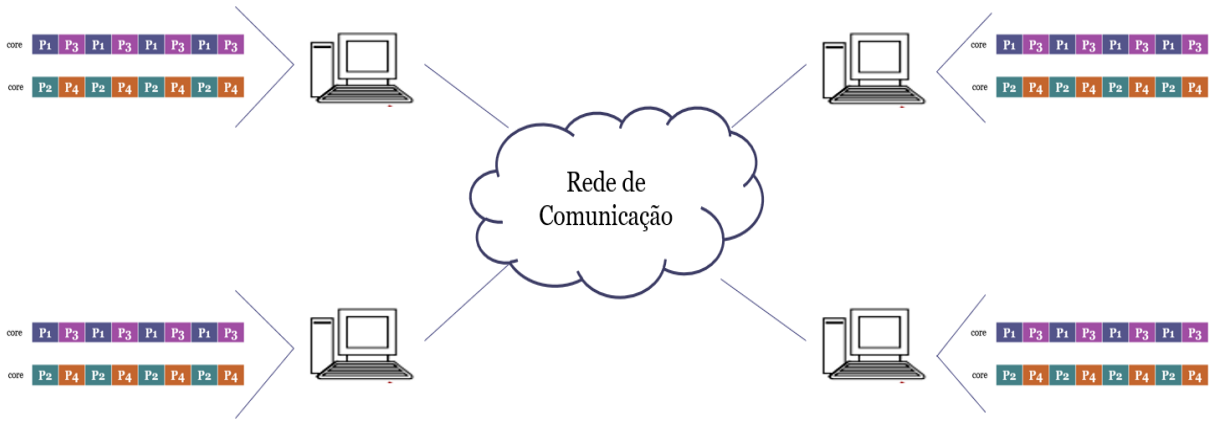

Figura 2-11 – Computação Paralela sendo executada em um Sistema Distribuído de computadores

Para o presente trabalho, a plataforma de serviços para Computação em Nuvem *Amazon Web Service* (AWS) foi escolhida como ambiente para executar as aplicações em um ambiente distribuído. A escolha se deu devido às características adequadas aos testes necessários do projeto e ao suporte da *Amazon* que disponibilizou gratuitamente um ambiente distribuído de hardware, softwares e sistemas de armazenamento para que as aplicações pudessem ser executadas. Conceitos relacionados a essa plataforma serão abordados no tópico a seguir.

2.8.1 *Amazon Web Service* - AWS

A *Amazon Web Services* (AWS) é uma plataforma de computação em nuvem desenvolvida e disponibilizada pela empresa multinacional Amazon. A plataforma oferece serviços de computação em nuvem como[9]:

- Infraestrutura como serviço (IaaS);
    - Contém os componentes básicos da TI em nuvem e, geralmente, dá acesso (virtual ou no hardware dedicado) a recursos de rede e computadores, como também espaço para o armazenamento de dados.
- Plataforma como serviço (PaaS);

[9] Disponível em https://aws.amazon.com/pt/types-of-cloud-computing/, acessado em 03 de abril de 2022.

- Permite que os desenvolvedores não precisam mais gerenciar a infraestrutura subjacente (geralmente, hardware e sistemas operacionais), permitindo que o foco do desenvolvimento seja na implantação e no gerenciamento das aplicações.

- Pacote de softwares como serviço (SaaS).
  - Possibilita que não seja necessário destinar recursos para definir como o serviço será mantido ou como a infraestrutura subjacente será gerenciada permitindo focar apenas na forma como o software deverá ser usado.

Atualmente, a AWS é a plataforma de nuvem mais usada em toda a rede mundial de computadores, oferecendo dezenas de serviços e instâncias[10] com preços dinâmicos baseados na oferta e demanda de recursos do mercado de instâncias (DUTRA, 2018; LIN; PAN; LIU, 2022). Nos tópicos a seguir serão detalhados os serviços da AWS usados no presente trabalho.

2.8.1.1 Amazon Elastic Compute Cloud – EC2

O *Amazon Elastic Compute Cloud* (*Amazon* EC2), pode ser considerado o serviço mais importante de toda AWS fornecendo ambientes virtualizados de acordo com a necessidade do projeto, podendo variar de servidores com capacidade mínima utilizados em desenvolvimento e testes, até maquinas com alto poder computacional para suportar aplicações em produção (DUTRA, 2018).

O *Amazon* EC2 oferece uma plataforma de computação ampla com mais de 500 instâncias e opções de processador, armazenamento, redes e sistemas operacionais para auxiliar o desenvolvedor de acordo com as necessidades das aplicações[11].

[10] Instâncias consistem em várias combinações de CPU, memória, armazenamento e capacidade de rede. Cada tipo de instância inclui um ou mais tamanhos de instância, permitindo que os recursos computacionais sejam alocados de acordo com os requisitos da carga de trabalho a ser executada. Disponível em https://aws.amazon.com/pt/ec2/instance-types/, acessado em 03 de abril de 2022.

[11] Disponível em https://aws.amazon.com/pt/ec2/instance-types/, acessado em 14 de abril de 2022.

2.8.1.2 Amazon Elastic MapReduce – EMR

O *Amazon Elastic MapReduce* (*Amazon* EMR) é uma plataforma para lidar com grande quantidade de dados em nuvem. Permite a execução de trabalhos de processamento de dados distribuídos em grande escala, consultas SQL interativas e aplicações de Aprendizado de Máquina. Faz uso dos *frameworks* de análise de código aberto como *Apache Spark* e *Apache Hive*, dentre outros[12].

Para o presente trabalho, o Amazon EMR foi usado como base para a execução das aplicações permitindo a configurações de *Clusters*[13] com instâncias executoras do EC2 (chamadas de *workloaders*) e permitindo também a execução de um ambiente com *Apache Spark* onde aplicações em linguagens Scala e Java (usadas no trabalho) pudessem ser executadas.

2.8.1.3 Amazon Simple Storage Service – S3

O *Amazon Simple Storage Service* (*Amazon* S3) é um serviço de armazenamento de objetos que oferece disponibilidade de dados e segurança para empresas e desenvolvedores. É possível no *Amazon* S3 armazenar e proteger dados de praticamente qualquer caso de uso, como aplicações nativas da nuvem e aplicações móveis.

No presente trabalho, o *Amazon* S3 foi usado como repositório para os arquivos de dados necessários para as execuções do modelo desenvolvido.

2.8.2 Métricas para Avaliação em Computação Paralela

A aplicação da Computação Paralela em determinados problemas computacionais está diretamente relacionada ao desempenho. Especificamente no contexto da Computação Paralela aplicada à problemas que envolvem Algoritmos Evolutivos, várias medidas podem ser usadas

[12] Disponível em https://aws.amazon.com/pt/emr/, acessado em 15 de abril de 2022.
[13] Conjunto de máquinas ligadas em rede para execução de aplicações de forma distribuída

para avaliar o comportamento das aplicações. Sudholt (2015) destaca métricas que podem ser usadas para avaliar Algoritmos Evolutivos nesse contexto. São elas o:

- *Speedup*
- *Efficiency*

O *Speedup* está relacionado a chamada Lei de Amdahl (AMDAHL, 1967) que leva o nome do arquiteto computacional Gene Amdahl. A Lei de Amdahl diz respeito a identificação de possíveis ganhos de desempenho ao adicionar cores de computação adicionais a aplicações que possui componentes seriais e paralelos (SILBERCHATZ; GALVIN; GAGNE, 2012). De acordo com a Lei de Amdahl, independente de quantos processadores são dedicados a execução da parte paralelizável de um programa, o tempo de execução mínima não pode ser menor que a parte do programa que não pode ser paralelizável (parte serial). O *Speedup*, portanto, é uma medida do grau de desempenho e mede a razão entre o tempo de execução sequencial e o tempo de execução em paralelo. A fórmula do *Speedup* é:

$$Speedup(m) = \frac{T(1)}{T(m)}$$

onde $Speedup(m)$ é o *Speedup* para m processadores, $T(1)$ é o tempo de execução para um processador (execução sequencial) e $T(m)$ é o tempo de execução para m processadores (execução paralela).

A *Efficiency* (ou Eficiência) mede o ganho de desempenho em relação ao consumo de recursos computacionais empregados na execução paralela (EAGER, ZAHORJAN; LAZOWSKA, 1989). A fórmula da Eficiência é:

$$E(m) = \frac{S(m)}{m} = \frac{T(1)}{m \times T(m)}$$

onde $E(m)$ é a Eficiência para m processadores, $S(m)$ é o *Speedup* para m processadores, $T(1)$ é o tempo de execução para um processador e $T(m)$ é o tempo de execução para m processadores.

As métricas *Speedup* e *Efficiency* serão usadas no presente trabalho para medir o grau de desempenho dos algoritmos nos testes realizados. O *Speedup* usando será o chamado *Weak Speedup Orthodox* definido em Sudholt (2015). Nessa métrica é realizada, segundo Sudholt (2015), uma comparação entre o Algoritmo Evolutivo executando em *m* maquinas e o mesmo Algoritmo Evolutivo executando em uma única máquina.

2.9 Considerações Finais

Nesse capítulo foram abordados os conceitos fundamentais que envolvem os Sistemas *Fuzzy* e os Algoritmos Genéticos, em especial, os Algoritmos Genéticos Multiobjetivos, com destaque para os algoritmos NSGA-DO, MNSGA-DO e NSGA-II. Questões como histórico, terminologias e características gerais dos sistemas que podem ser formados com essas técnicas foram tratadas. Abordou-se também questões relacionadas à Computação Paralela como um meio para possibilitar o processamento de grandes conjuntos de dados. Outro tópico abordado foi o relacionado ao modelo de programação *MapReduce* e ao *framework Apache Spark*.

Os conceitos abordados nesse capítulo formam a base para o desenvolvimento do projeto de abordagem paralela para geração de Bases de Regras *Fuzzy* em problemas que envolvem grandes conjuntos de dados.

Capítulo 3

ARTIGOS RELACIONADOS

3.1 Considerações Iniciais

Neste capítulo serão apresentados os artigos relacionados ao presente projeto abordando trabalhos que lidam com Algoritmos Genéticos, Sistemas *Fuzzy*, Computação Paralela e Problemas que envolvem dados com grande volume e a alta dimensionalidade.

Por fim, seguem as considerações finais abordando uma síntese dos temas abordados e apresentando o foco do presente trabalho.

3.2 Sistemas *Fuzzy* Genéticos Multiobjetivo

A Lógica *Fuzzy* e os Algoritmos Genéticos são técnicas de IA que podem atuar separadas ou em conjunto na resolução de vários problemas. A Lógica *Fuzzy*, em particular, é bastante usada em projetos envolvendo áreas como engenharia (elétrica, mecânica, mecatrônica, aeroespacial), matemática, desenvolvimento de softwares, pesquisas médicas, ciências sociais dentre outras áreas (SINGH *et al.*, 2013).

Já os Algoritmos Genéticos podem ser usados para encontrar as melhores soluções para problemas complexos em vários domínios como biologia, engenharia, ciência da computação, ciência sociais, dentre outros (KUMAR *et al.*, 2010).

Em conjunto, essas 2 técnicas podem ser usadas visando a obtenção de melhores resultados na resolução de problemas. Essa união forma os chamados Sistemas *Fuzzy* Genéticos.

Como exemplo do uso dos Sistemas *Fuzzy* Genéticos, pode-se citar o trabalho de Herrera, Lozano e Verdegay (1994) onde é demonstrado que o uso de AG em conjunto com os sistemas *Fuzzy* propicia o tratamento da incerteza incorporando flexibilidade a problemas de otimização e busca.

Outro exemplo do uso dos Sistemas *Fuzzy* em conjunto com os Algoritmos Genéticos pode ser visto no trabalho de Alcalá-Fdez, Alcalá e Herrera (2011). Nesse trabalho, tido como uma das principais referências na área de Sistemas de Classificação Baseados em Regras *Fuzzy* (SCBRF), quando há um aumento no número de variáveis do problema há o aumento exponencial no espaço de busca e, consequentemente, das regras *Fuzzy* obtidas. É dito que esse crescimento dificulta o processo de aprendizagem e pode levar a problemas de tempo e memória consumidos além de problemas relacionados à complexidade (em relação ao número de regras obtidas e ao número de variáveis incluídas em cada regra).

c propõem então um método de classificação baseado em regras de associação *Fuzzy* com seleção de regras genéticas para problemas de alta dimensão. O objetivo foi o de obter um classificador baseado em regras *Fuzzy* preciso e compacto com baixo custo computacional. Foi então desenvolvido o sistema de classificação baseada em regras de associação *Fuzzy* para problemas de alta dimensão (FARC-HD) com seleção de regras genéticas. Nesse processo os autores usaram um sistema de classificação associativa construído em dois estágios. O primeiro estágio foi desenvolvido para descobrir as regras de associação inerentes ao banco de dados. O segundo estágio foi usado para selecionar um subconjunto de regras para construir um classificador. Esse subconjunto de regras era constituído apenas de regras que possuíam no máximo 3 antecedentes. A partir desse subconjunto, era então extraídas combinações que formariam as BRF finais.

Alcalá-Fdez, Alcalá e Herrera (2011) concluíram que o modelo obteve o melhor desempenho no estudo experimental e apresentou um baixo custo computacional em todos os conjuntos de dados, obtendo um bom desempenho. Destacou-se que a proposta obteve modelos com um número reduzido de regras com poucos atributos no antecedente, dando a vantagem de alta interpretabilidade do ponto de vista do usuário.

Outros tipos de problemas abordados com o uso dos Sistemas *Fuzzy* Genéticos são aqueles em que, não apenas uma, mas um conjunto de varáveis devem ser otimizadas. Esses problemas são caracterizados como problemas de otimização multiobjetivo. Os SFG criados para otimizar mais de uma varável são chamados de Sistemas *Fuzzy* Genéticos Multiobjetivos. Nesse caso, a diferença se dá pela busca não de apenas uma solução, mas de um conjunto de soluções para a resolução de um problema.

Na otimização multiobjetivos são considerados, em determinados casos, objetivos conflitantes entre si, ou seja, quando melhoramos um deles, em geral, pioramos o outro. Para a resolução desse tipo de problema o conceito de Dominância de Pareto (explicado com mais detalhes no Capítulo 2, tópico 2.4 - Otimização Multiobjetivo, pág. 39) é comumente adotado.

Nesse contexto, pode-se citar o trabalho de Cárdenas e Camargo (2012) onde foi desenvolvido um método genético multiobjetivo para aprender regras *Fuzzy* e otimizar conjuntos *Fuzzy* em SCBRF. A ideia era realizar a busca do equilíbrio entre 2 objetivos: Acurácia e Interpretabilidade. Para isso foram realizados três estágios sequenciais:

- Definição de Base de Dados
- Aprendizado de Base de Regras e
- Otimização da Base de Dados

O trabalho foi realizado a partir da implementação de duas técnicas de otimização multiobjetivo: NSGA-II (do inglês "*Non-dominated Sorting Genetic Algorithm II*") de Deb *et al.*, (2002) e o SPEA2 (do inglês "*Strenght Pareto Evolutionary Algorithm 2*") de Zitzler, Laumanns e Thiele (2001). Nesse método a melhor regra a ser inserida na Base de Regras em cada iteração é selecionada entre as soluções não dominadas usando um critério relacionado à acurácia da Base de Regras. Os experimentos demonstraram que o método melhorou a acurácia no segundo estágio quando comparado a outros três métodos genéticos mantendo uma interpretabilidade equivalente medida pelo tamanho do modelo. No terceiro estágio, o modelo proposto em Cárdenas e Camargo (2012) melhora a interpretabilidade em relação à semântica dos conjuntos *Fuzzy* mantendo a precisão em níveis adequados.

Em Cárdenas e Camargo (2013) foi apresentado uma versão melhorada do método de Cárdenas e Camargo (2012) visando aprender regras de classificação *Fuzzy* a partir de dados por meio de um AEMO. Na nova versão do método, um novo critério para selecionar a melhor regra é usado. Nesse novo método considera-se a interpretabilidade semântica no nível da base

de regras, no caso, especificamente o número de regras disparadas (diferentemente do método anterior que usa o critério relacionado à acurácia da base de regras). A nova versão do método proposto alcançou resultados equivalentes aos da versão anterior com relação à acurácia, entretanto houve melhora tanto na interpretabilidade semântica no nível de base de regras quanto no número de regras disparadas.

Ainda no contexto da otimização multiobjetivo, tem-se o trabalho de Fazzolari, *et al.* (2013) onde é apresentada a visão geral dos Sistemas *Fuzzy* Evolutivos Multiobjetivos descrevendo as principais contribuições da área. Nesse trabalho, são apresentadas algumas sugestões de problemas que podem ser investigados. Temas como avaliação de performance de SFEMO e automatização do processo de escolha de soluções foram algum dos temas abordados.

Dentre os tópicos abordados em Fazzolari, *et al.* (2013), pode-se destacar o tema que envolvem dados com grande volume, alta dimensionalidade em um contexto de SCBRF.

Fazzolari, *et al.* (2013) destacam que problemas relacionados a conjuntos de dados grandes e de alta dimensão ocorrem cada vez mais e que o tratamento das questões que envolvem esses sistemas nesse contexto mostra-se como um campo de investigação interessante.

Fernández *et al.* (2015) afirmam que no tratamento de grande quantidade de dados há um aumento no tempo computacional despendido para tratamento dos dados. Já em Elkano *et al.* (2017) é dito que a principal desvantagem dos SCBRF é a queda de desempenho que ocorre quando do processamento de grandes quantidades de dados.

Outros trabalhos como os de Lopez *et al.* (2014), Ferranti *et al.* (2017), Rodriguez-Mier, Mucientes e Bugarín (2019), Fernandez-Basso, Ruiz e Martin-Bautista (2021), Bhukya e Sadanandam (2022) e Aghaeipoor, Javidi e Fernandez (2022) também fazem referências aos Sistemas *Fuzzy* e aos Sistemas Genéticos. Esses trabalhos serão abordados no tópico a seguir.

3.3 Computação Paralela no Contexto dos SFG

A Computação Paralela surge, no contexto dos Sistemas *Fuzzy* Genéticos, como um meio para propiciar o devido processamento dos dados em trabalhos voltados para a temática de grandes conjuntos de dados.

Em Deb (2015) salienta-se que, com o uso de técnicas de computação paralela, pode-se obter a Fronteira de Pareto Ótima em problemas que envolvem os AEMO. Neste caso, cada processador em um ambiente de computação paralelo recebe um único espaço de busca e, no final de cada procedimento computacional, as várias partes são unidas para encontrar a Fronteira de Pareto Ótima completa.

Em Sudholt (2015) são destacados os efeitos da paralelização, ressaltando a capacidade da abordagem em proporcionar a redução do tempo computacional ao passo que, ao mesmo tempo, conduz ao aumento da exploração do espaço de busca. Outro ponto destacado é a melhoria da diversidade obtida na população de um AG quando comparado com algoritmos evolutivos sequenciais.

Luna e Alba (2015) apoiam Sudholt (2015) ao destacarem que o método de paralelizar algoritmos evolutivos surge como uma forma de reduzir o tempo computacional à valores aceitáveis ao passo que permite o uso da paralelização como meio para melhorar a diversidade populacional.

Outros trabalhos, como os de Fernandez, Rio e Herrera (2016), Rodríguez-Fdez, Mucientes e Bugarín(2016a), Ferranti *et al.* (2017), Fernandez, Almansa e Herrera (2017), Elkano *et al.* (2018, 2019) Fernandez-Basso, Ruiz e Martin-Bautista (2021) e Bhukya e Sadanandam (2022) usam os *frameworks Apache Hadoop* e *Apache Spark* como meio para construção e aperfeiçoamento de SFBR no contexto de problemas de problemas que envolvem dados com grande volume e/ou alta dimensionalidade.

Nesse cenário, essas aplicações de computação paralela para construção e aperfeiçoamento de SFBR, mostram-se promissoras, uma vez que possibilitam o processamento dos dados a partir do particionamento e do processamento separados deles.

Um exemplo é o trabalho de Lopez *et al.* (2014) onde é apresentada a proposta de um Sistemas de Classificação Baseada em Regras *Fuzzy* capaz de fornecer um modelo interpretável e acurado no cenário de grande volume de dados. O sistema foi denominado de *Chi-FRBCS-BigData* e foi baseado no artigo de Chi, Yan e Pham (1996). Nesse artigo foi desenvolvido um método para lidar com problemas de grande volume de dados a partir do método clássico de SCBRF.

O algoritmo do *Chi-FRBCS-BigData* usa dois processos *MapReduce* diferentes para lidar com duas partes diferentes do algoritmo.

O primeiro processo *MapReduce* é dedicado à construção da Base de Conhecimento *Fuzzy* de um conjunto de treinamento de *Big Data*. O segundo processo *MapReduce* é usado para predizer a classe de amostras pertencentes a conjuntos grandes amostras de dados.

O processo de construção da BC *Fuzzy*, que pode ser visto na Figura 3-1, é iniciado a partir da criação da Base de Dados *Fuzzy* com o uso de funções de associação triangular igualmente distribuídas. Em seguida, o sistema segmenta automaticamente o conjunto de dados de treinamento original em blocos de dados independentes que são transferidos automaticamente para as diferentes unidades de processamento juntamente com a base de dados *Fuzzy* criada.

Na Figura 3-1 pode ser vista a fase de *MAP*. Essa fase ocorre na medida em que cada unidade de processamento trabalha independentemente sobre os dados disponíveis para construir sua Base de Regras *Fuzzy*. A seguir, cada processo do *MAP* pesquisará regras com o mesmo antecedente. Se as regras compartilham o mesmo resultado, apenas uma regra é preservada; se as regras tiverem consequentes diferentes, somente a regra com o maior peso é mantida na BR *Fuzzy*.

Ainda na Figura 3-1, pode ser vista a fase de *REDUCE* que ocorre quando os resultados obtidos por cada processo de mapeamento são combinados para formar o BRF final. Nessa fase, regras contraditórias (regras com o mesmo antecedente, com ou sem o mesmo consequente e com peso de regra diferente) podem ter sido criadas. Nesse caso, procedimentos específicos para lidar com essas regras contraditórias são executados. Esses procedimentos definem duas variantes do algoritmo *Chi-FRBCS-BigData*.

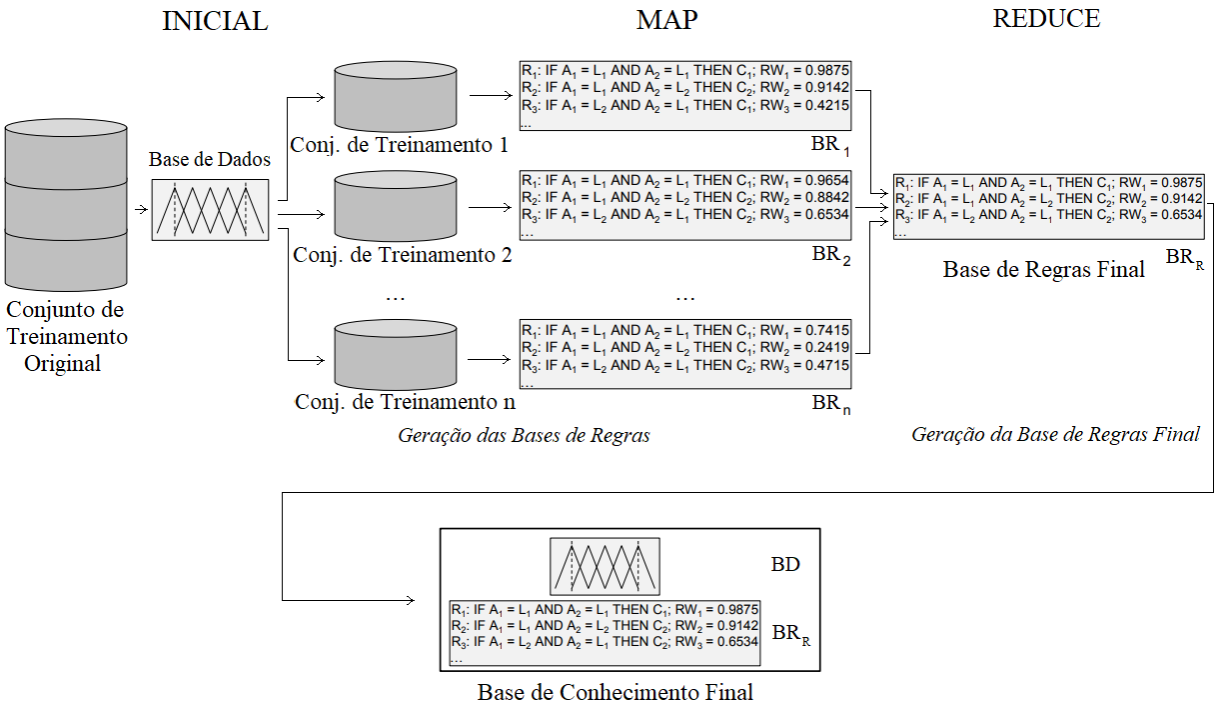

Figura 3-1 - Fluxograma da fase de construção do BC no algoritmo Chi-FRBCS-BigData (adaptado de Lopez *et al.* (2014))

Na primeira versão, chamada de *Chi-FRBCS-Bigdata-Max*, o método procura as regras com o mesmo antecedente. Neste caso, apenas a regra com o maior peso é mantida na BRF final. Nessa situação, não é necessário verificar se o consequente é o mesmo ou não, pois serão mantidas apenas as regras com maior peso. Lopez *et al.* (2014) destacam que regras equivalentes (regras com o mesmo antecedente e consequente) podem possuir pesos diferentes pois foram computadas em diferentes processos e em diferentes conjuntos de treinamento.

Na segunda versão, chamada de *Chi-FRBCS-Bigdata-Ave*, o método também procura as regras com o mesmo antecedente. No entanto, o peso médio das regras que têm o mesmo consequente é calculado e a regra com o maior peso médio é mantida na BRF final. Essa etapa é necessária, pois, como dito anteriormente, regras com o mesmo antecedente e consequente podem ter pesos diferentes por terem sido construídas em conjuntos de treinamento distintos.

Para demonstrar a diferença entre os algoritmos, é apresentado em Lopez *et al.* (2014) um exemplo com cinco regras com o mesmo antecedente e com os respectivos pesos apresentados na Tabela 3-1.

Tabela 3-1 – Exemplo de regras com o mesmo consequente

Regra	Consequente	Peso
R_1	Classe 1	0,8743
R_2	Classe 2	0,9254
R_3	Classe 1	0,7142
R_4	Classe 2	0,2143
R_5	Classe 1	0,8215

Para o algoritmo *Chi-FRBCS-Bigdata-Max* a regra mantida será a Regra R_2 (consequente "*Classe 2*"), pois está é a que apresenta o maior peso ($PesoR_2 = 0{,}9254$).

No caso do algoritmo *Chi-FRBCS-Bigdata-Ave*, o peso médio das regras com o mesmo consequente é calculado. Nesse caso o cálculo é o seguinte:

$$Classe\ 1 = PesoR_1 + PesoR_3 + PesoR_5 = [0{,}8743 + 0{,}7142 + 0{,}8215] \div 3 = 0{,}8033$$

$$Classe\ 2 = PesoR_2 + PesoR_4 = [0{,}9254 + 0{,}2143] \div 2 = 0{,}5699$$

O algoritmo *Chi-FRBCS-Bigdata-Ave* define, portanto, a permanência da regra com consequente "*Classe 1*", pois esta é a regra com maior peso médio ($Peso\ médio = 0{,}8033$).

Na fase *FINAL*, os resultados computados nas fases anteriores são fornecidos como saída do processo de cálculo. Assim, a Base de Conhecimento gerada será composta pela Base de Dados concebida na fase "*INICIAL*" e a Base de Regras obtida na fase "*REDUCE*".

A BC obtida será usada para prever a classe dos novos exemplos. Na Figura 3-2, Lopez *et al.* (2014) demonstram como são estimadas as classes dos novos dados nas 3 diferentes fases chamadas de "*INICIAL*", "*MAP*" e "*FINAL*".

Na fase *INICIAL* o sistema segmenta automaticamente o conjunto de dados de *Big Data* original em blocos de dados independentes. Esses blocos são transferidos para as diferentes unidades de processamento junto com a BC *Fuzzy* criada anteriormente.

Na fase de *MAP*, cada função estima a classe para os exemplos incluídos em sua partição de dados. Nesse caso, cada unidade de processamento passa por todos os exemplos do seu respectivo conjunto de dados e prevê a classe de saída de acordo com a BC *Fuzzy*.

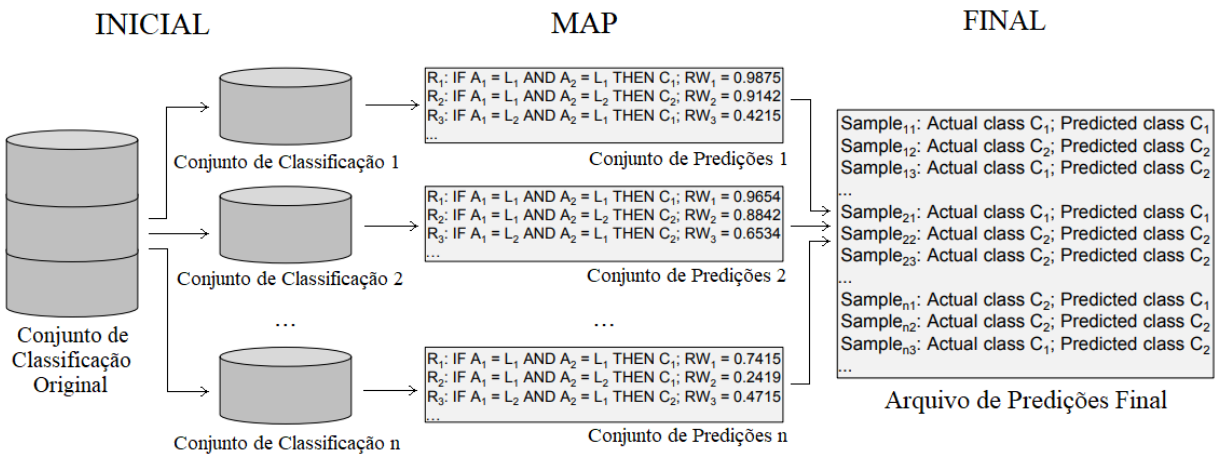

Figura 3-2 - Fluxograma da fase de estimativa da classe dos dados no algoritmo Chi-FRBCS-BigData (adaptado de Lopez et al. (2014))

Na fase *FINAL*, os resultados calculados na fase anterior são fornecidos como a saída do processo de cálculo. As classes estimadas para os diferentes exemplos do conjunto de classificação de *Big Data* são agregadas apenas concatenando os resultados fornecidos por cada função da fase de *MAP*.

Lopez *et al.* (2014) afirmam que os resultados dos testes realizados corroboram a eficiência da abordagem proposta. É destacado, no entanto, que não é possível afirmar se o método *Chi-FRBCS-BigData-Max* é ou não mais adequado do que o método *Chi-FRBCS-BigData-Ave*. É dito que, quando se tem um menor número de divisões dos dados na fase de *MAP*, o método *Chi-FRBCS-BigData-Ave* tende a ser o mais adequado. Em caso de grande divisão dos dados na fase de *MAP* com foco em resultados mais rápidos, o método *Chi-FRBCS-BigData-Max* tende a ser a melhor escolha.

Para o trabalho aqui proposto, os testes revelaram uma situação semelhante à relatada em Lopez *et al.* (2014). A quantidade de partições impostas aos dados no início do processamento impacta diretamente no tempo de processamento e nos resultados obtidos. Percebeu-se que, no presente projeto, particionar em demasia gera resultados mais rápidos, porém pode levar a uma homogeneidade das subpopulações obtidas em cada partição fazendo com que os indivíduos obtidos durante as gerações do AG não sejam capazes de explorar todo o espaço de busca de forma satisfatória gerando, assim, BRF pouco acuradas. Em contrapartida, reduzir demasiadamente a quantidade de partições leva a geração de indivíduos mais heterogêneos entre si e, consequentemente, com maior capacidade de explorar os espaços de

buscas que, nesse caso, são maiores em cada partição. Assim, devido a essa heterogeneidade das populações, aumenta-se a probabilidade de se encontrar melhores soluções. Contudo, destaca-se que essa redução na quantidade de partições leva há um maior tempo de processamento dos dados, pois mais avaliações deverão ser feitas no processo de cálculo do *Fitness* de cada indivíduo.

Outro trabalho que usa a abordagem *MapReduce* para o contexto de grande volume de dados é o de Fernandez, Rio e Herrera (2016). Nesse trabalho é usado o modelo de programação *MapReduce* com o intuito de abordar um grande volume de dados de forma eficiente proporcionando um melhor desempenho ao modelo gerado na forma de SFBR. É destacado que problemas de classificação baseados em regras *Fuzzy* tem mostrado bons resultados para a abordagem *MapReduce* em problemas que envolvem grande quantidade de dados afirmando que os sistemas criados podem ser melhorados quando usados em conjunto com algoritmos evolutivos.

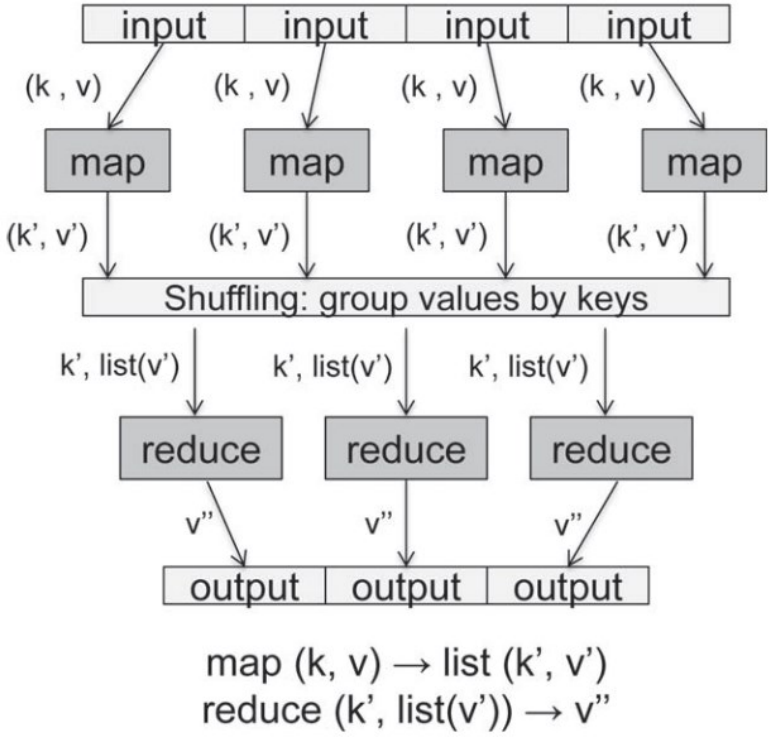

Figura 3-3 - O Modelo *MapReduce* (extraído de Fernandez, Rio e Herrera (2016))

O modelo *MapReduce* apresentado em Fernandez, Rio e Herrera (2016) pode ser visto na Figura 3-3. Nela, é possível ver a fase de *MAP* onde os dados divididos são coletados e, posteriormente, alimentam a função *REDUCE*. A seguir, uma função é aplicada e se obtém os

resultados finais. Nesse modelo, é usado uma estrutura de dados essenciais conhecida como par <*key*, *value*>. Na Figura 3-3, *k* e *v* referem-se ao par chave original e valor respectivamente; *k'* e *v'* formam o par intermediário criado depois da função de *MAP*; e *v''* é o resultado final do algoritmo.

A computação paralela é usada também no trabalho de Rodríguez-Fdez, Mucientes e Bugarín (2016a) onde é apresentado um Sistema *Fuzzy* Genético chamado de S-FRULER[14], que é uma versão escalável do FRULER[15] proposto em Rodríguez-Fdez, Mucientes e Bugarín, (2016b).

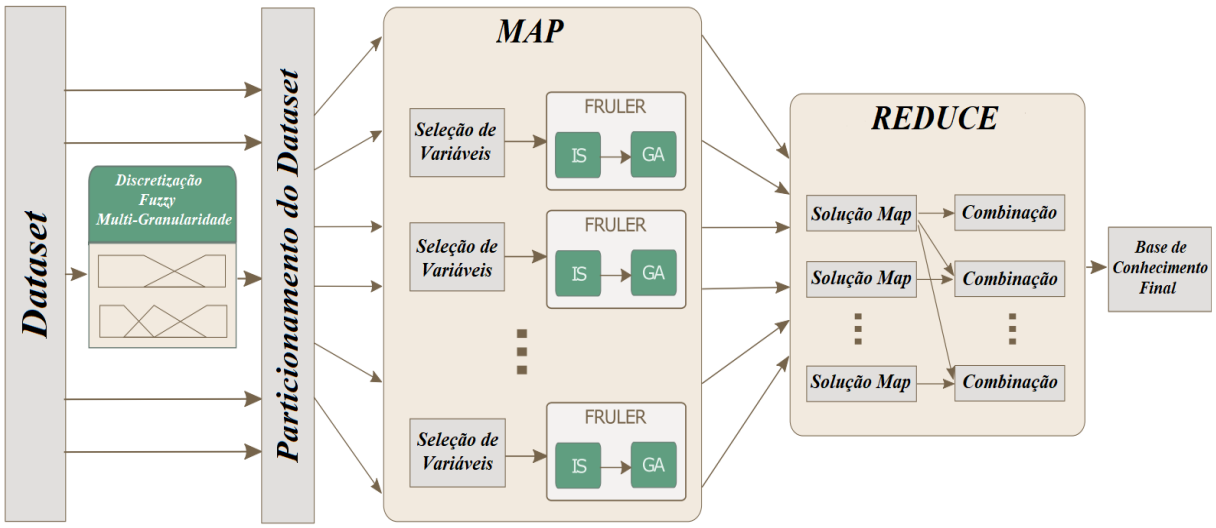

Figura 3-4 - Arquitetura S-FRULER com as fases de pré-processamento, de MAP e de REDUCE (adaptado de Rodríguez-Fdez, Mucientes e Bugarín (2016a))

O S-FRULER usa a computação paralela em problemas de regressão possibilitando a obtenção de modelos com alta acurácia e baixa complexidade (menor número de regras e menor número de antecedentes das regras) enquanto o tempo de execução do algoritmo é reduzido.

Com o S-FRULER é possível manter as características de acurácia e interpretabilidade (complexidade) do FRULER quando da aplicação em problemas de grande escala. A

[14] S-FRULER é um acrônimo para *Scalable Fuzzy Rule Learning Through Evolution for Regression*. O S-FRULER é uma versão escalável do FRULER.

[15] FRULER é um acrônimo para *Fuzzy Rule Learning Through Evolution for Regression*. O FRULER é um sistema para aprendizado de regras *Fuzzy Takagi-Sugeno-Kang* por intermédio de técnicas evolutivas para problemas de regressão.

arquitetura proposta em Rodríguez-Fdez, Mucientes e Bugarín (2016a) pode ser vista na Figura 3-4.

Na Figura 3-4 é apresentada a arquitetura S-FRULER onde o problema de regressão é dividido em partes menores ao incorporar um processo de seleção de recursos para diminuir o número de varáveis em cada uma das partes. Dessa forma, cada parte do problema é repassada e resolvida independentemente no bloco chamado de *MAP*. Posteriormente, o bloco *REDUCE*, recebe as partes e obtém a Base de Conhecimento Final (que é composta da Base de Dados e da Base de Regras *Fuzzy*) a partir da informação gerada em cada partição do problema. Assim, as questões de desempenho podem ser resolvidas mesmo que seja considerado o contexto relacionado a problemas com grande quantidade de dados.

Outro trabalho que usa computação distribuída no contexto de SFEMO é o de Ferranti, Marcelloni e Segatori (2016). Nele, é proposta uma versão distribuída do algoritmo multiobjetivo PAES (*Pareto Archived Evolution Strategy*) para aprendizado simultâneo das bases de dados e de regras de Classificadores Baseados em Regras *Fuzzy* (CBRF) com o objetivo de maximizar a precisão e minimizar a complexidade.

O PAES foi desenvolvido inicialmente no trabalho Mann e Smith (1996) e teve suas versões iniciais usadas na comparação com um Algoritmo Genético Multiobjetivo no trabalho de Knowles e Corne (1999).

No AEMO PAES básico, inicialmente é gerada uma solução aleatória ***S*** e ocorre, então, uma mutação ***M*** a partir dessa solução. Caso ***S*** domine ***M***, então ***M*** é descartada. Caso ***M*** domine a solução ***S***, então ***M*** é adicionada a um arquivo externo e o processo reinicia. Caso ***M*** não seja dominada por nenhum membro do arquivo externo, então é usada uma função para verificar quão densa estão as regiões onde se encontram as soluções. Isso permite decidir qual a melhor solução a ser adicionada ao conjunto de soluções final. O processo é repetido até que um determinado critério de parada seja atingido.

Especificamente no trabalho de Ferranti, Marcelloni e Segatori (2016), a versão distribuída do PAES, chamada de DPAES, foi implementada usando o *framework Apache Spark*.

Ferranti, Marcelloni e Segatori (2016) destacam que, apesar de outros trabalhos terem tido bons resultados em termos de desempenho e do tratamento de problemas com grande volume de dados ao usar o *Apache Hadoop*, existem trabalhos, como o de Zhou (2010), onde

os custos extras em relação às operações de entrada e saída do sistema de arquivos distribuídos e a sobrecarga de contabilidade do sistema reduziram significativamente os benefícios do paralelismo no *framework Hadoop*. Ferranti, Marcelloni e Segatori (2016) usaram então o *Apache Spark* como estrutura de processamento de dados.

A versão distribuída do PAES consistiu de 2 fases principais intituladas:

- Fase 1 – Geração de Regras Candidatas Distribuídas
- Fase 2 – Otimização Evolutiva Distribuída

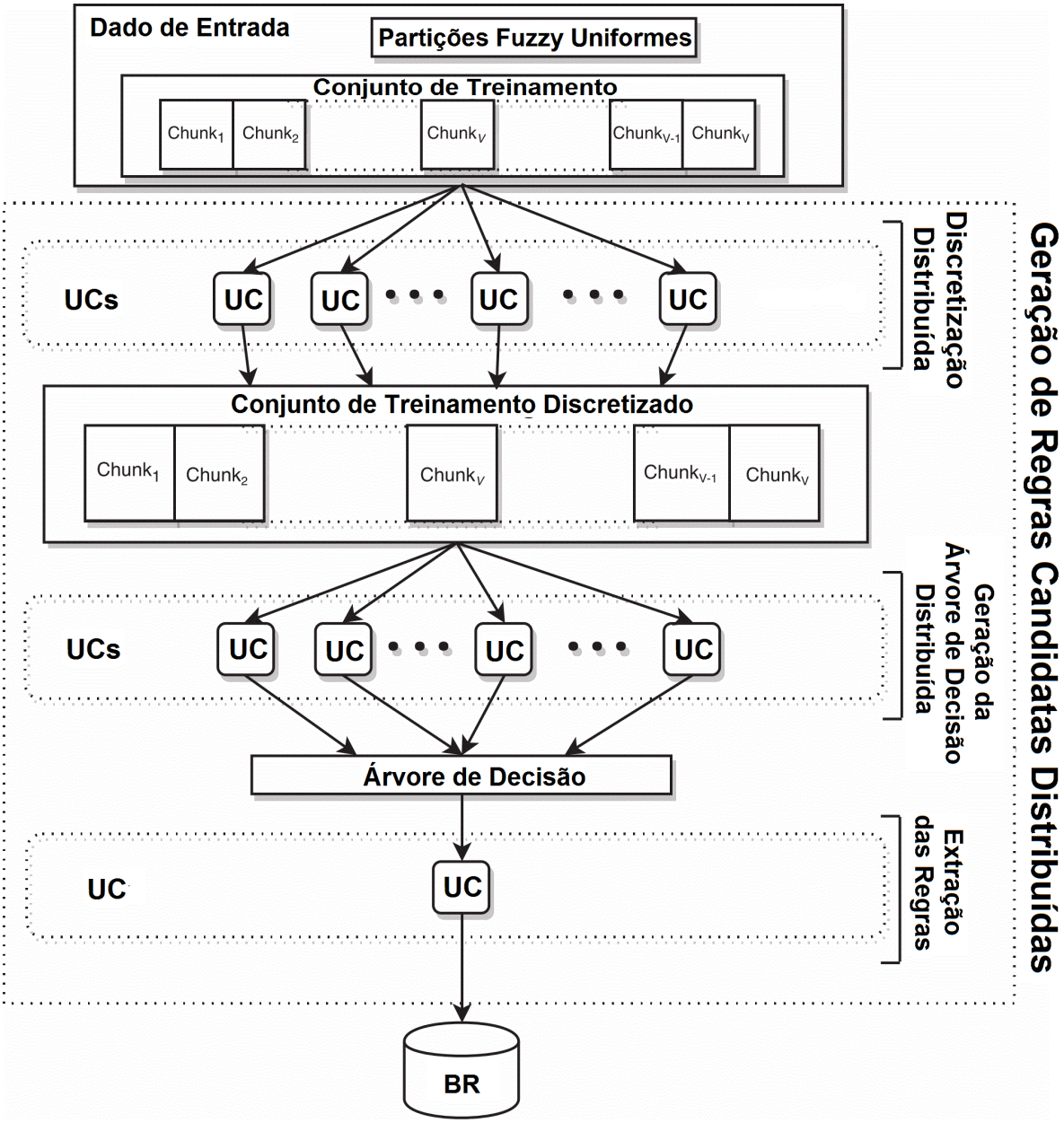

Figura 3-5 – Fase 1 – Geração de Regras Candidatas Distribuídas (Adaptado de Ferranti, Marcelloni e Segatori (2016))

Na Fase 1, tem-se 3 etapas como pode ser visto na Figura 3-5. Na primeira (Discretização Distribuída), cada variável de entrada contínua é discretizada usando uma partição uniforme *Fuzzy* com 5 conjuntos.

Para cada *Chunk* (partição do conjunto de dados de treinamento) é associado um valor categórico correspondente ao valor do conjunto com maior grau de pertinência. Essa discretização é aplicada em paralelo usando as chamadas Unidades de Computação (UC). Na segunda etapa (Geração da Árvore de Decisão Distribuída) uma árvore de decisão distribuída C4.5 modificada é executada no conjunto de treinamento discretizado.

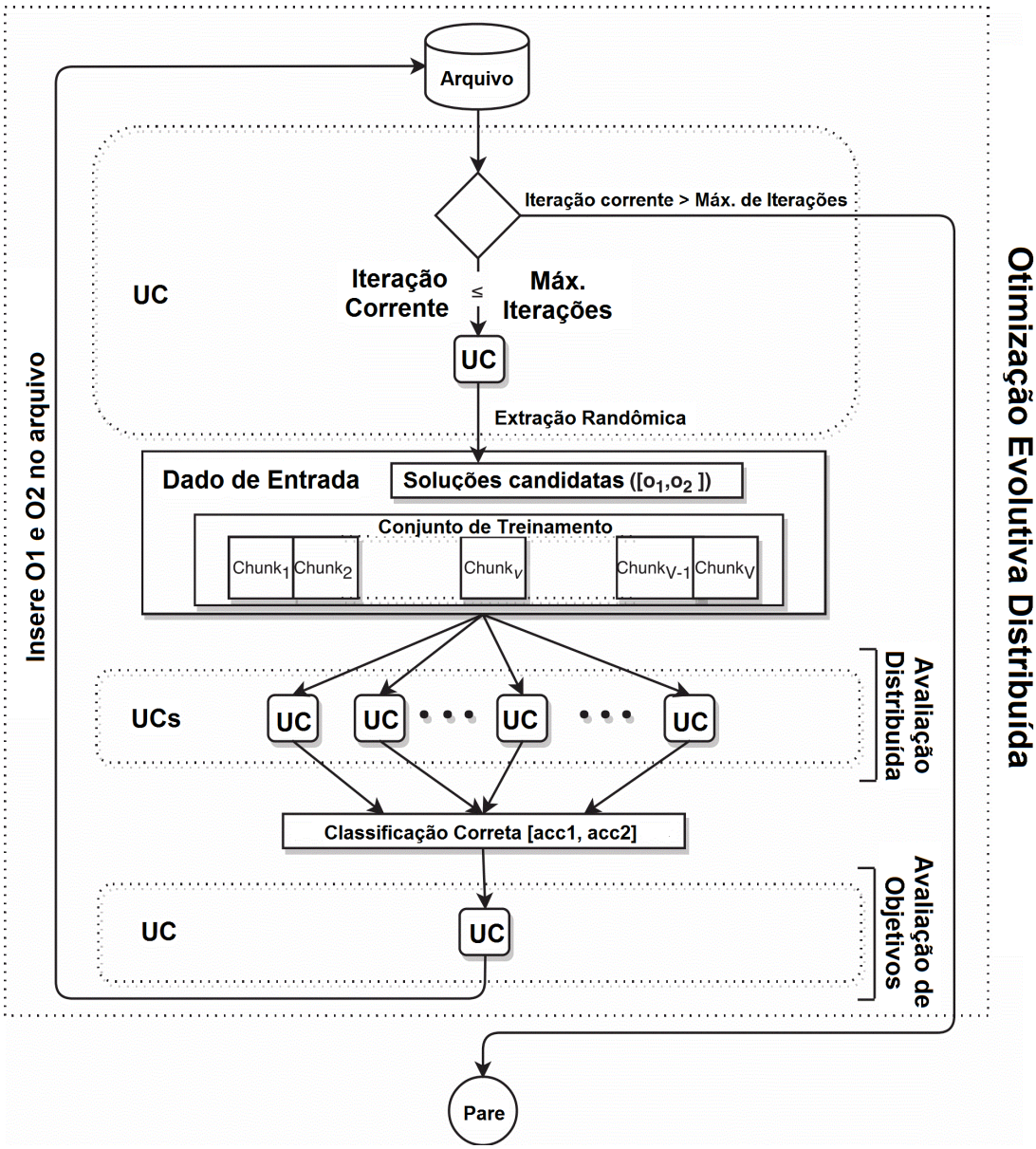

Figura 3-6 - Fase 2 - Otimização Evolutiva Distribuída (Adaptado de Ferranti, Marcelloni e Segatori (2016))

Com isso, a base de regras é extraída da árvore de decisão e encerra-se a Fase 1 do processo. Nesse ponto, são geradas, aleatoriamente, duas soluções que são adicionadas ao arquivo do DPAES para dar início à Fase 2, como pode ser visto na Figura 3-6.

Na segunda fase, o cálculo da taxa de classificação em cada iteração do AEMO é distribuído adotando o modelo mestre-escravo. Detalhadamente, são geradas duas soluções candidatas o_1 e o_2 ao se aplicar os operadores genéticos às soluções s_1 e s_2 extraídas aleatoriamente do arquivo do DPAES. Para cada *Chunk* do conjunto de treinamento, ambas as soluções candidatas (o_1 e o_2) classificam todas as instâncias pertencentes àquele *Chunk*.

Na Figura 3-6, as variáveis acc1 e acc2 correspondem aos contadores associados às soluções candidatas o_1 e o_2 e guardam a quantidade de vezes que houve classificação correta.

Em seguida, é verificado se as soluções candidatas o_1 e o_2 serão adicionadas ao arquivo. Elas serão adicionadas somente se não forem dominadas por nenhuma solução contida no arquivo. As soluções que passarem a ser dominadas por o_1 e o_2 são removidas do arquivo. O processo global é executado até que o número de máximo de avaliações de aptidão seja alcançado e a Fase 2 é então encerrada.

Testes foram realizados a fim de verificar a eficiência da proposta em termos de acurácia e complexidade, além de avaliar o aumento de velocidade alcançado em um *cluster* de computadores. Foi demonstrado nesse trabalho a capacidade de extrair, com eficiência, bases de regras compactas e com alta acurácia mesmo com a manipulação de grandes conjuntos de dados em hardware considerado pelo artigo como ‘modesto’.

Ainda em Ferranti, Marcelloni e Segatori (2016), percebe-se que, na arquitetura proposta, as duas soluções extraídas do arquivo do DPAES são usadas em todas os *Chunks* de uma determinada iteração. Esse fato limita o poder do paralelismo que pode ser alcançado em relação ao AEMO. Nessa arquitetura, seria possível, em uma mesma iteração, selecionar várias duplas de soluções candidatas, para serem usadas pelas UC no processamento das instâncias. Desse modo, baseado na classificação obtida por cada par de soluções, seria possível tanto determinar qual dessas soluções iria ou não para o arquivo, quanto definir, baseado nas classificações obtidas pelas soluções anteriores, qual o critério que seria usado para selecionar as próximas soluções a serem usadas nas próximas iterações.

Outro ponto a ser destacado é o fato de o modelo *main-worker*[16] ter sido adotado como forma de propiciar a distribuição do cálculo da classificação. Esse método provoca um aumento do tempo de execução do algoritmo uma vez que requer a varredura do conjunto de treinamento global impactando na velocidade de execução da abordagem proposta. Esse fato é destacado no próprio artigo e uma alternativa seria o desenvolvimento de um método que abandonasse a classificação de uma UC que estivesse processando um *Chunk* e demorasse em demasia para executar sua tarefa. Esse fato poderia acelerar a execução do algoritmo uma vez que evitaria que tarefas que já haviam sido executadas esperassem muito tempo por um processo em execução. No entanto, o fato de abandonar determinados *Chunks* pode, por vezes, conduzir o processo a soluções que não sejam próximas às soluções esperadas.

O uso da abordagem paralela no contexto de problemas de *Big Data* também é abordado no trabalho Elkano *et al.* (2018). Nesse artigo é introduzido uma solução *MapReduce* para o projeto de um Sistemas de Classificação Baseados em Regras *Fuzzy* voltado para problemas de *Big Data*. O sistema desenvolvido, denominado de CHI-BD, permite gerar, no contexto de *Big Data*, o mesmo modelo que o obtido pelo algoritmo original proposto em Chi, Yan e Pham (1996) caso esse pudesse ser executado com essa mesma quantidade de dados.

O modelo de Chi, Yan e Pham (1996) (no artigo de Elkano *et al.* (2018) é chamado de CHI_{LOCAL}^{BD}) depende fortemente do número de *MAPPERS*[17] usados para execução do algoritmo. É destacado que isso ocorre por dois motivos:

1) Cada *MAPPER* calcula os pesos das regras considerando apenas um subconjunto do conjunto de treinamento. Esse fato faz com que a qualidade dos pesos das regras dependa da distribuição do subconjunto especificado. Quanto mais nós são adicionados, menor a qualidade dos pesos das regras.

2) O tamanho de amostra afeta os pesos das regras. Em casos onde o número de *MAPPERS* é maior que o número de instancias de uma classe, haverá *MAPPERS*

[16] Nomenclatura usada para designará que em um sistema distribuído há um computador principal ligado a vários computadores que executarão de forma distribuída uma determinada aplicação.

[17] Os *MAPPERS são os n*ós de computação responsáveis pelo processamento dos dados na fase de *Map* do modelo *MapReduce.*

sem representação da classe quando se tem conjuntos de dados binários. Isso implica que os pesos das regras gerados nesses *MAPPERS* serão insignificantes com todos os exemplos pertencendo à mesma classe. Além disso, essas regras serão mantidas na base de regras final, pois possuem o peso máximo possível.

Devido a esses motivos, a ideia do trabalho de Elkano *et al.* (2018) foi projetar uma solução computacional que gera exatamente o mesmo modelo, independentemente da configuração dos *clusters*. Nesse caso, é possível garantir que o grau de paralelismo adotado não afeta o desempenho do algoritmo.

O método desenvolvido foi chamado de CHI^{BD}_{GLOBAL} e consiste de dois estágios diferentes. O primeiro estágio está relacionado ao processo de geração de regras e o segundo relacionado ao cálculo dos pesos das regras.

Na primeira etapa, todas as regras *Fuzzy* são criadas sem calcular seus pesos, obtendo uma base de regras preliminar. Para esse fim, uma nova regra é gerada para cada instância em cada *MAPPER*. Ao fim do processo, *REDUCERS*[18] agrupam automaticamente as regras fornecidas pelos *MAPPERS*, obtendo uma lista de todos os possíveis consequentes (lista de consequentes) para cada regra (antecedentes).

Na segunda etapa, a base de regras obtida na primeira etapa é compartilhada por todos os nós, o que permite calcular os pesos das regras usando todas as instâncias. Para isso, em cada *MAPPER*, o grau de correspondência de todas as regras com as instâncias no *MAPPER* é calculado. Posteriormente, todos esses graus correspondentes são usados pelos *REDUCERS* para calcular o peso final de cada regra a fim de decidir, automaticamente, o consequente final na lista de classes.

Os resultados do experimentos realizados em Elkano *et al.* (2018) mostraram que, ao lidar com problemas de grande volume de dados, o método CHI^{BD}_{GLOBAL} (método proposto no

[18] O *REDUCER* (componente da fase de *Reduce* do modelo *MapReduce*) responsável por agregar as saídas dos *MAPPERS*.

artigo) supera o método CHI_{LOCAL}^{BD} (abordagem clássica proposta em Chi, Yan e Pham (1996)) em termos de tempo de execução e desempenho de classificação.

É também destacado que, como o Chi original calcula os pesos das regras calculando o grau de correspondência de cada regra com todos os exemplos, é necessário lidar com a questão da complexidade do tempo. Nesse caso, o aumento no tamanho dos dados tem um efeito não-linear no tempo de execução o que limita a expansão do algoritmo. Para resolver esse problema, Elkano *et al.* (2018) sugerem um processo de seleção prévia de *features* (características) que levaria a um aumento na probabilidade de regras duplicadas provocando, por consequência, uma redução no tamanho da base de regras. Outra opção sugerida é a definição de um *threshold* (limitante) que remova as regras geradas por poucos exemplos.

Outro trabalho que relaciona os Sistemas *Fuzzy* com problemas de grande quantidade de dados é o de Elkano *et al.* (2019). Nesse trabalho é destacado que a interpretabilidade sempre foi uma grande preocupação para os classificadores baseados em regras *Fuzzy*. Entretanto, é dito que, em problemas de classificação que envolvem grande quantidade de dados, os Sistemas de Classificação Baseados em Regras *Fuzzy* não conseguem manter o bom equilíbrio entre a acurácia e a interpretabilidade que caracterizou esses mesmos sistemas em ambientes que não envolvem toda essa gama de dados.

Elkano *et al.* (2019) afirmam que os métodos mais acurados constroem modelos muito complexos (grande número de regras e de conjuntos *Fuzzy*) enquanto abordagens focadas na interpretabilidade não fornecem uma acurácia adequada.

Com base nessas questões, Elkano *et al.* (2019) propõem um novo algoritmo de aprendizado distribuído chamado CFM-BD voltado para construir SCBRF acuradas e compactas para o contexto de problemas que envolvem grande quantidade de dados e que não adapta ou estende nenhum método existente.

O processo desenvolvido é composto de 3 fases. São elas:

1) Pré-processamento baseado no teorema da transformação integral de probabilidade
    - Nessa fase são criados conjuntos *Fuzzy* (rótulos linguísticos) que se ajustem à distribuição real dos dados de treinamento, mantendo constante o número de conjuntos *Fuzzy* por variável (por exemplo, baixo, médio, alto). Aqui, os conjuntos *Fuzzy* são construídos e os dados de treinamento são pré-processados

(distribuição original dos dados de treinamento é transformada em uma distribuição uniforme).

2) Indução de regras inspirada nos algoritmos CHI-BD e *Apriori*;
    - Nessa fase a base de regras é construída aplicando um algoritmo de indução de regras projetado para grande volume de dados. Esse processo consiste em dois estágios sequenciais, inspirados em alguns dos conceitos introduzidos nos algoritmos CHI-BD de Elkano *et al. (*2018) e *Apriori* de Agrawal e Srikant (1994). Em uma primeira etapa um conjunto de itens frequentes mais promissores é extraído e, em uma segunda etapa, a base de regras *Fuzzy* é criada.

3) Seleção de regras por meio de uma otimização evolutiva global.
    - Nessa etapa um processo de seleção de regras é realizado para obter um modelo preciso e compacto. Para isso, é aplicado o algoritmo evolutivo de Eshelman (1991). Foi implementada uma nova versão distribuída desse algoritmo evolutivo que executa um processo de otimização global que avalia a qualidade de cada indivíduo considerando todo o conjunto de treinamento.

É destacado no artigo que as fases processam todo o conjunto de dados de maneira distribuída fazendo com que o modelo obtido não seja afetado pela distribuição das partições e pelo grau de paralelismo.

Foi realizado um estudo empírico completo para testar o desempenho da abordagem CFM-BD proposta. Acurácia, complexidade e tempo de execução foram analisados e os resultados obtidos foram comparados com 3 classificadores *Fuzzy* para *Big Data* (**Chi-Spark-RS** de Fernandez, Almansa e Herrera (2017), **FBDT**(*Fuzzy Binary Decision Trees*) / **FMDT** (*Fuzzy Multiway Decision Trees)***,** de Segatori, Marcelloni e Pedrycz (2018) e **CHI-BD** de Elkano *et al.* (2018)).

De acordo com este estudo, o CFM-BD mostrou-se competitivo em relação aos classificadores comparados usando, no entanto, modelos mais simples compostos por poucas regras e que possuíam menos de três antecedentes e com cinco rótulos linguísticos para as variáveis. Nos testes, o CFM-BD criou algumas regras compostas por menos de 3 antecedentes, enquanto os outros métodos geram milhares de regras contendo muitos antecedentes e rótulos

linguísticos, com exceção de alguns modelos do FBDT que constroem, geralmente, menos de 32 regras. No entanto, esses modelos são geralmente menos acurados que o CFM-BD e não alcançaram o desempenho tido como de "estado-da-arte". Destacou-se também que, nos testes, o CFM-BD mostrou-se competitivo, além da interpretabilidade, também em termos de desempenho de classificação.

Outro trabalho que faz uso da Computação Paralela no contexto dos SF é o trabalho de Fernandez-Basso, Ruiz e Martin-Bautista (2021). Nesse trabalho são propostos diferentes algoritmos de mineração de regras de associação *Fuzzy* no âmbito de grande quantidade de dados para a descoberta de padrões de concorrência a partir de conjuntos de dados *Fuzzy*. No trabalho foi demonstrado que algoritmos não distribuídos propostos para mineração de regras de associação *Fuzzy* podem falhar ao manipular conjuntos de dados massivos (dados com grande volume) devido a erros de estouro de memória. Esse fato faz com que a eficiência do sistema seja afetada a medida o conjunto de dados aumenta.

Com o uso da Computação Paralela e do *framework Apache Spark* foi possível então, processar os dados massivos demonstrado que com o uso dessas técnicas é possível obter ganhos em relação ao tempo de execução, ao uso da memória e em relação à capacidade de processamento dos sistemas propostos.

Há também o trabalho de Bhukya, Sadanandam (2022) onde uma técnica foi desenvolvida para lidar com grande volume de dados no ambiente *Hadoop MapReduce*. Nesse trabalho, é projetado um *Cluster* com modelo de geração de regras para classificação de dados em ambiente *Hadoop MapReduce* possibilitando a manipulação dos conjuntos de dados de valor contínuo, onde o ponto de dados não oferece nenhum detalhe de classe e pode ser incerto. O objetivo da técnica desenvolvida é, portanto, produzir um conjunto de regras ótimo para o processo de classificação de dados. Além disso, a técnica desenvolvida é usada para uma classificação de grande volume de dados e engloba os processos de agrupamento e classificação obtendo resultados satisfatórios de acurácia nos experimentos.

Outro trabalho é o de Aghaeipoor, Javidi e Fernandez (2022), onde é proposto o IFC-BD, um classificador *Fuzzy* interpretável para lidar com grande volume de dados. O objetivo é de, por meio do aprendizado de um modelo *Fuzzy* compacto e preciso, tornar o sistema mais interpretável. O IFC-BD é desenvolvido em uma estrutura distribuída baseada em células através dos três estágios de trabalho de aprendizado inicial de regras, generalização de regras e seleção de regras heurísticas. Todo este procedimento permite ir desde um número elevado de

regras específicas até um número menor de regras mais gerais e seguras. Além disso, para resolver possíveis conflitos de regras, um novo peso estimado de regras é proposto especificamente para problemas de que envolvem grande quantidade de dados. O IFC-BD foi avaliado em comparação com as abordagens de última geração do paradigma de classificação *Fuzzy*, considerando interpretabilidade, acurácia e tempo de execução. Os resultados dos experimentos revelaram que o algoritmo proposto foi capaz de melhorar a interpretabilidade de classificadores baseados em regras *Fuzzy*, bem como seu desempenho preditivo.

Como pode-se perceber, diversos trabalhos fazem uso da Computação Paralela no contexto de SFBR e de Algoritmos Genéticos a fim de extrair, de dados com grande volume e/ou alta dimensionalidade regras para formar uma Base de Regras *Fuzzy* que seja devidamente acurada e interpretável.

Na Tabela 3-2 encontra-se uma sumarização das abordagens para tratamento de problemas que envolvem dados com grande volume e/ou alta dimensionalidade.

Tabela 3-2 - Sumarização das abordagens para tratamento de problemas que envolvem dados com grande volume e/ou alta dimensionalidade

Autor	Ano	Nome	Foco	Algoritmo Evolutivo Base	Framework	Linguagem	Tipo de problema
Alcalá-Fdez, Alcalá e Herrera	2011	FARC-HD	Acurácia e Interpretabilidade	CHC evolutionary algorithm (Eshelman (1991))	-	Java	Classificação
Lopez et al.	2014	Chi-FRBCS-BigData	Acurácia	Não usa	Apache Hadoop	Java	Classificação
Rodríguez-Fdez, Mucientes e Bugarín	2016	S-FRULER	Acurácia e Interpretabilidade	Algoritmo evolutivo não nomeado	Apache Spark	Java	Regressão
Fernandez, Rio e Herrera	2016	Chi-EFS2T-BigData	Acurácia	CHC evolutionary algorithm (Eshelman (1991))	Apache Hadoop	Java	Classificação
Ferranti et al.	2017	DPAES-RCS	Acurácia e Interpretabilidade	PAES (Mann e Smith (1996))	Apache Spark	Java	Classificação
Fernandez, Almansa e Herrera	2017	Chi-Spark-RS	Acurácia e Interpretabilidade	CHC evolutionary algorithm (Eshelman (1991))	Apache Spark	Scala	Classificação
Elkano et al.	2018	CHI-BD	Acurácia	Não usa	Apache Hadoop	Java	Classificação
Elkano et al.	2019	CFM-BD	Acurácia e Interpretabilidade	CHC evolutionary algorithm (Eshelman (1991))	Apache Spark	Scala	Classificação
Fernandez-Basso, Ruiz e Martin-Bautista	2021	BDFARE	Speedup e Eficiência	Não usa	Apache Spark	Python (numpy)	Classificação
Bhukya, Sadanandam	2022	OC-RSRGM	Acurácia	Não usa	Apache Hadoop	Java	Classificação
Aghaeipoor, Javidi e Fernández	2022	IFC-BD	Acurácia e Interpretabilidade	C evolutionary algorithm (Eshelman (1991))	Apache Spark	Scala	Classificação

3.4 Aplicações no Contexto dos SFG

Diversos trabalhos tem usado os SFG em aplicações que envolvem áreas como engenharias, matemática, desenvolvimento de softwares, pesquisas médicas, ciências sociais dentre outras áreas (SINGH *et al.*, 2013).

Um desses trabalhos é o de Jamshidi e Pilevar (2013), que usa os sistemas *Fuzzy* em conjunto com os Algoritmos Genéticos, para realizar a automatização da segmentação de imagens de ressonância magnética para estimar o número correto de segmentos. Nessa pesquisa, o método *Fuzzy*-Genético foi testado em imagens de ressonância magnética e obteve melhor resultados do que os métodos tradicionais. Números corretos de segmentos foram obtidos automaticamente levando à convergência para solução desejada em um menor número de gerações.

Em Park *et al.* (2014) é apresentado um método para medir o desempenho de sistema de manufatura. Este trabalho combina AG e um método de análise de dados a fim de selecionar o melhor cenário para alocação de operador. O método proposto gera, por meio do GA, um conjunto de cenários de alocação de operador e posteriormente esses cenários são avaliados. Como as variáveis do sistema podem ser imprecisas, a Lógica *Fuzzy* é usada para auxiliar na classificação dos dados e na escolha do melhor cenário. Os resultados obtidos a partir de um estudo de caso prático são apresentados e ilustram a eficácia do uso dos AG atuando em conjunto com os Sistemas *Fuzzy*.

Outros trabalhos também vêm sendo realizados visando aplicar os conhecimentos desenvolvidos nas pesquisas de SFG em problemas cotidianos. Esse foi o caso do trabalho de Rodriguez-Mier, Mucientes e Bugarín (2019). Nesse trabalho foi apresentada a abordagem baseada em um sistema Genético *Fuzzy*, para construir bases de conhecimento acuradas e interpretáveis para prever o consumo de energia em edifícios inteligentes. Para realizar essa tarefa, foi proposto um sistema de computação cognitiva para previsão de várias etapas tendo como base o S-FRULER de Rodríguez-Fdez, Mucientes e Bugarín (2016a) explicado anteriormente nesse capítulo. No trabalho o S-FRULER juntamente com um método de seleção de subconjunto de recursos foi usado em um edifício da Universidade de Santiago de Compostela na Espanha. O edifício em questão continha mais de 450 sensores e atuadores que

monitoraram e controlaram diferentes partes do prédio. Foi destacado que um dos principais problemas foi a geração de modelos *Fuzzy* interpretáveis e acurados em tempo razoável, dada a grande quantidade de dados gerados no edifício. Esse fato corroborou com o uso de técnicas escalonáveis para lidar com o aumento da complexidade. Os experimentos com dados reais sobre dois problemas diferentes relacionados ao gerenciamento de energia revelaram uma melhoria média de 6% na acurácia em relação ao S-FRULER sem seleção de características e com bases de conhecimento com um número menor de variáveis. Além disso, as regras geradas foram consideradas interpretáveis o que permitiu tomadas de decisões relacionadas à redução do consumo de energia.

3.5 Pesquisas do Laboratório CIG-Departamento de Computação-UFSCar

Os pesquisadores associados ao laboratório CIG (*Computational Inteligence Group*) do Departamento de Computação da Universidade Federal de São Carlos – UFSCar – São Paulo – Brasil desenvolvem projetos na área de Sistemas *Fuzzy* e Algoritmos Genéticos. Diversos trabalhos foram desenvolvidos e alguns deles, com relação direta com o presente trabalho, serão descritos a seguir.

Nesse contexto, é apesentada na Figura 3-7 a cronologia de alguns dos projetos desenvolvidos aos logo dos anos no laboratório CIG da UFSCar.

Em 2012 foi publicado o *artigo* "*Using fuzzy formal concepts in the genetic generation of fuzzy systems*" (CINTRA; MONARD; CAMARGO, 2012) e posteriormente, em 2016, o trabalho foi estendido e publicado sob o título de "*Genetic generation of fuzzy systems with rule extraction using formal concept analysis*"(CINTRA; CAMARGO; MONARD, 2016). Nesses trabalhos buscou-se introduzir uma nova abordagem para a geração genética de sistemas *Fuzzy*. A novidade da proposta, denominada Método FCA-*Based*, está na combinação hibrida de Conceitos Formais *Fuzzy* (do inglês *Fuzzy Formal Comcepts* - FCA) para extrair regras para formar o espaço de busca de um AG. A ideia central é a de que o método FCA-*Based* possibilita a geração automática de BRF a partir de um conjunto de regras obtidos por meio da teoria formal de análise de conceitos diretamente dos dados.

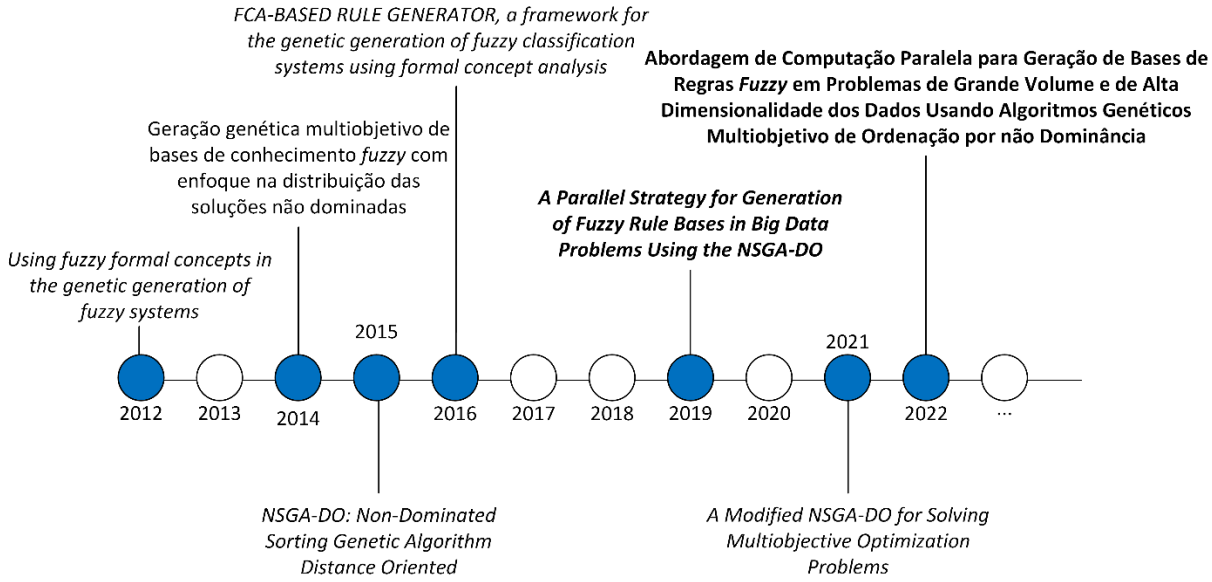

Figura 3-7 - Cronologia dos projetos voltados para o desenvolvimento de Sistemas *Fuzzy* e Sistemas Genéticos do CIG - UFSCar

No método FCA-*Based*, após a extração das regras que formam o espaço de busca genética, usa-se um AG para selecionar a base de regras final. Em uma última etapa, o método proposto realiza uma poda de regras para melhorar a interpretabilidade das bases de regras *Fuzzy*. A extração de regras proposta para o algoritmo baseado em FCA apresenta complexidade polinomial e não requer a predefinição do número de regras a serem extraídas a partir dos dados. Como extrai regras diretamente dos dados, o método FCA-*Based* evita a extração aleatória de regras. Apresenta também a vantagem de extrair automaticamente regras com número variável de condições em seus antecedentes. Além disso, um método de seleção de subconjunto de características, projetado especificamente para sistemas de classificação *Fuzzy*, é integrado ao método baseado em FCA para reduzir o espaço de busca de soluções.

O método FCA-*Based* foi comparado a oito diferentes SFBR e os resultados dos experimentos demostraram que o método proposto alcançou uma maior acurácia com sete dos oito métodos comparados. Na Figura 3-8 é apresentado um esquema gráfico de como se relacionam os módulos do método FCA-*Based*.

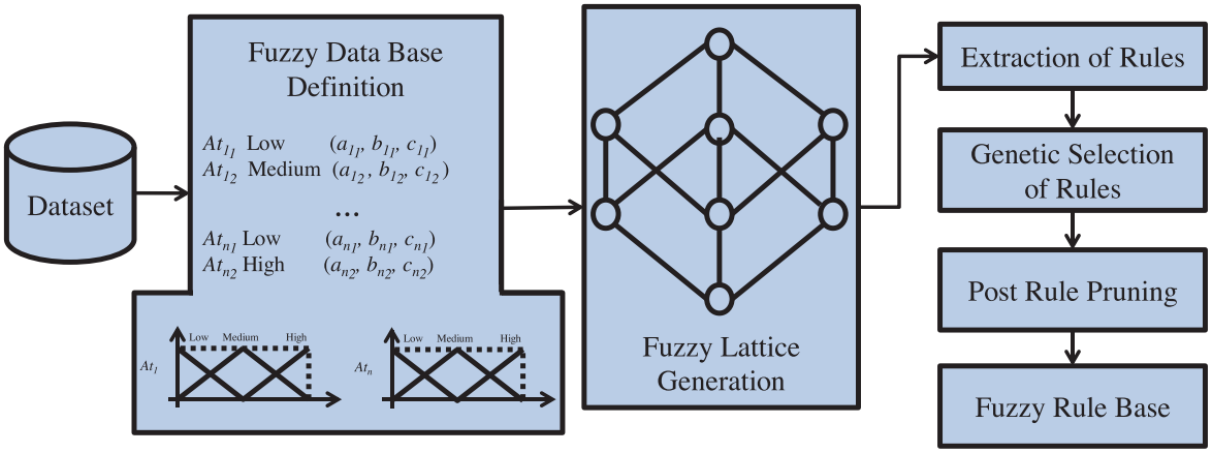

Figura 3-8 - Método FCA-*Based* (CINTRA; CAMARGO; MONARD, 2016)

Com os dados disponíveis, a base de dados *Fuzzy* é definida (quantidade e disposição dos conjuntos *Fuzzy*) e então tem-se a fase chamada de *Fuzzy Lattice Generation*. Nessa fase, os dados são representados em forma de tabelas onde as colunas representam os atributos e as linhas representam os objetos. Em outras palavras, no contexto formal há a representação da relação entre objetos e seus atributos. Após essa fase, há a extração das regras e uma posterior seleção de regras por meio da aplicação de um Algoritmo Genético Mono-objetivo. A seguir, tem-se uma fase chamada de *Post Rule Pruning.* Nessa fase ocorre a poda de algumas regras a fim de melhorar a interpretabilidade da BRF final. Na etapa de *Post Rule Pruning,* o processo de seleção de regras primeiro ordena, para cada classe, as regras de acordo com o número de condições. Posteriormente, se uma regra mais específica do que outra regra for encontrada, a regra mais específica será removida do conjunto de regras se a taxa de erro do classificador sem essa regra não aumentar. Este processo é repetido até que todas as regras sejam testadas e a BRF final seja definida.

Em paralelo, um outro trabalho estava sendo desenvolvido voltado ao desenvolvimento de um AGMO com foco na distribuição das soluções na fronteira de Pareto. Em 2014 surgiu então o trabalho "Geração genética multiobjetivo de bases de conhecimento *Fuzzy* com enfoque na distribuição das soluções não dominadas" (PIMENTA, 2014) e posteriormente, em 2015, um artigo foi publicado sob o título de "*NSGA-DO: Non-Dominated Sorting Genetic Algorithm Distance Oriented*" (PIMENTA; CAMARGO, 2015).

Nesse projeto foi desenvolvido o NSGA-DO (do inglês "*Non-dominated Sorting Genetic Algorithm Distance Oriented*"), a partir de uma modificação do conhecido NSGA-II

(do inglês "*Non-dominated Sorting Genetic Algorithm II*") de Deb et al., (2002). A ideia era que o algoritmo proposto fosse capaz de encontrar soluções não dominadas que equilibrassem a fronteira de Pareto no que diz respeito à otimização dos objetivos. A principal característica do NSGA-DO é a seleção de soluções orientada à distância. Assim, a cada iteração, as soluções não dominadas são usadas para encontrar uma aproximação para a fronteira de Pareto. O algoritmo utiliza as localizações das soluções na fronteira aproximada para encontrar a melhor distribuição de soluções, o que orienta as operações de seleção do AGMO. Uma descrição mais detalhada do NSGA-DO pode ser vista no Capítulo 2 – Fundamentação Teórica – seção 2.4.1.1 – *Nondominated Sorting Genetic Algorithms*, na pág. 41. Para validar a proposta, o NSGA-DO, foi aplicado ao contexto de SFEMO para a geração de bases de conhecimento *Fuzzy* para classificação.

No projeto, usou-se as BRF obtidas pelo método FCA-*Based* e, por meio de uma seleção genética (com o uso do AGMO NSGA-DO ao invés do AG Mono-objetivo do projeto FCA-*Based* original), obteve-se BRF superiores às encontradas com o uso do NSGA-II nas três questões analisadas: dispersão de soluções não dominadas, precisão e interpretabilidade dos sistemas gerados.

Outro projeto desenvolvido nessa linha foi o trabalho de 2021 publicado com o título de "*A Modified NSGA-DO for Solving Multiobjective Optimization Problems*" (MACHADO *et al.*, 2021). Esse trabalho foi desenvolvido em uma parceria entre o laboratório CIG da UFSCar – SP e o laboratório LASIC (Laboratório de Sistemas Inteligentes e Cognitivos) da Universidade Estadual de Feira de Santana – UEFS – Bahia – Brasil.

O algoritmo MNSGA-DO publicado em Machado *et al.* (2021) foi apresentado como uma melhoria do processo de seleção do NSGA-DO. A mudança ocorre na quantidade de Pontos Ideais que são criados em relação ao NSGA-DO (o número de Pontos Ideias é igual a 2 vezes o número de soluções) e também em relação a forma como as soluções são associadas a esses pontos. No NSGA-DO, várias soluções podem ser aglutinadas em um mesmo ponto diferentemente do que ocorre no MNSGA-DO. Esse fato corrobora com o espalhamento maior das soluções ao longo da fronteira de Pareto. Os resultados demonstram que o MNSGA-DO, em um contexto de teste voltado à otimização de funções, superou o NSGA-II e o NSGA-DO em quase todos os *benchmarks*, obtendo soluções mais precisas e diversificadas.

Os projetos citados avançaram até esse ponto com resultados significativos nas áreas de SF e de AG. Todavia, quando se considera contextos onde os conjuntos de dados usados para extrair as regras pelo método FCA-*Based* levam a uma quantidade muito grande de regras a serem avaliadas e selecionadas, os métodos sequenciais propostos encontram dificuldades. Assim, quando há uma quantidade massiva de regras geradas, o processo de geração de bases de regras leva a problemas relacionados ao desempenho dos sistemas, ao custo computacional e relacionados ao tempo de execução. A partir dessa premissa, surgiu o trabalho de 2019 intitulado "*A Parallel Strategy for Generation of Fuzzy Rule Bases in Big Data Problems Using the NSGA-DO*" (SANTANA; CAMARGO, 2019).

Nesse trabalho foi proposta uma estratégia de modelagem paralela para a geração de BRF em problemas que envolvem dados com grande volume e alta dimensionalidade utilizando o AGMO NSGA-DO. A ideia é que fosse possível gerar BRF balanceando os objetivos de acurácia e interpretabilidade a partir da aplicação de uma abordagem de Computação Paralela e do uso de um AGMO.

A partir dessa ideia foi então concebido o presente projeto, cujo objetivo é desenvolver uma abordagem de Computação Paralela para geração de BRF em problemas com grande volume e com alta dimensionalidade dos dados. A ideia é realizar essa tarefa usando regras extraídas de conjuntos de dados a partir do Método FCA-*Based* e com o uso dos Algoritmos Genéticos Multiobjetivo de Ordenação por não Dominância NSGA-DO, MNSGA-DO e NSGA-II.

3.6 Considerações Finais

Neste capítulo, artigos relacionados aos temas de computação paralela e problemas que envolvem dados com grande volume e/ou alta dimensionalidade no contexto de SFG foram apresentados.

Foi possível perceber que a Lógica *Fuzzy* e os Algoritmos Genéticos atuando em conjunto, colaborativamente ou de forma híbrida, têm sido aplicadas em diversos trabalhos. Foi percebido também que problemas que envolvem grande quantidade de dados vêm sendo estudados e soluções voltadas para o tratamento da acurácia e da interpretabilidade das Regras

Fuzzy dos sistemas desenvolvidos nesse contexto vêm sedo desenvolvidas e aplicadas de variadas formas.

No presente trabalho a ideia é desenvolver uma abordagem que faz uso da Computação Paralela para geração de BRF em problemas que envolvem grande volume e alta dimensionalidade dos dados. Pretende-se usar os Algoritmos Genéticos Multiobjetivo de Ordenação por não-Dominância NSGA-DO, proposto em Pimenta e Camargo (2015) e MNSGA-DO, proposto em Machado *et al.* (2021) para extração de BRF obtidas de conjuntos de dados por meio do método FCA-*Based* proposto em Cintra, Camargo e Monard (2016). Mais detalhes da proposta serão abordados no próximo capítulo.

.

Capítulo 4

PROPOSTA

4.1 Considerações Iniciais

Nesse capítulo, será apresentada a proposta do trabalho, as etapas de desenvolvimento bem como as principais ferramentas usadas. Por fim, serão abordadas as considerações finais a respeito da proposta e o que se espera com a execução da mesma.

4.2 Proposta do Trabalho

Nos últimos anos, novas soluções foram desenvolvidas para lidar com problemas que envolvem dados com grande volume e alta dimensionalidade e, dentre elas, destacam-se aquelas que fazem uso dos Sistemas *Fuzzy* Genéticos em conjunto com a Computação Paralela no desenvolvimento de Sistemas *Fuzzy* Baseados em Regras.

Nesse sentido, a proposta no presente trabalho está no desenvolvimento de uma abordagem que faz uso da Computação Paralela para geração de Bases de Regras *Fuzzy* a partir de conjuntos de regras obtidos por meio da aplicação, em conjuntos de dados, do método de extração de regras FCA-*Based*. Esse projeto se insere no contexto de grande volume e alta dimensionalidade dos dados e faz uso dos Algoritmos Genéticos Multiobjetivo de Ordenação por não-Dominância NSGA-DO, MNSGA-DO e do clássico NSGA-II para buscar BRF ponderadas entre os objetivos conflitantes de acurácia e complexidade.

Para um melhor entendimento da proposta, serão apresentados, a seguir, mais detalhes do projeto.

4.3 Computação Paralela na Geração de BRF em Problemas com Grande Volume e Alta Dimensionalidade dos Dados

No caso do presente trabalho, as dificuldades de se processar todo um conjunto de dados são enfrentadas a partir da divisão das operações com o uso da Computação Paralela.

A arquitetura proposta pode ser vista na Figura 4-1. Na Fase A todo o conjunto de regras é dividido em *chunks* e cada um desses *chunks* é dividido em partições para serem processados paralelamente. A divisão dos *chunks* pode ser vista na fase B da Figura 4-1. Em seguida, cada um desses *chunks* é divido em partições de acordo com a quantidade de *Worloaders* (Unidades de Processamento – UP) disponíveis no sistema distribuído usado no processamento dos dados.

Em Zecevic e Bonaci (2017) é dito que definir o número de partições RDD é um passo importante porque, além de influenciar a distribuição de dados em todo o cluster, também determina diretamente o número de tarefas que executarão transformações RDD. Assim, se o número de partições for muito pequeno, o *cluster* será subutilizado. Além disso, podem ocorrer problemas de memória, pois os conjuntos de trabalho podem ficar grandes demais para serem carregados nas memórias das UP. É recomendado o uso de partições iguais a até quatro vezes o número de núcleos disponíveis no *cluster*. Entretanto, Zecevic e Bonaci (2017) salientam que valores moderadamente maiores, por vezes, não representam problemas e testes podem ser feitos para verificar o comportamento dos *Workloaders* quando o número de partições é aumentado acima desse valor.

Na fase C tem-se os *chunks* divididos em partições que deverão ser encaminhadas pelo Spark para serem executadas nas UP (fase D). Após as execuções das partições as UP retornam os resultados dos processamentos dos dados para a *Workloader* central (UP Principal) (fase E). Esses retornos são BRF extraídas a partir do processamento das regras de cada uma das partições. Essas BRF são unidas em um grande conjunto de regras (fase F) formado pela resposta obtida pela execução de cada um dos *Chunks* e de suas respectivas partições. Esse

conjunto obtido na fase F é então avaliado de acordo com o número de regras presentes e então é definido se ele deve ser reprocessado ou não. Caso seja verificado que ainda há um número grande de regras que deve ser reduzido, o conjunto obtido na fase F é dividido novamente em *chunks* de acordo com a quantidade de regras e a quantidade de nós disponíveis no chunks e então o processo se repete nas fases G, H, I e J até que um novo conjunto de regras K é obtido. Se esse conjunto K ainda for considerado grande de acordo com um determinado critério pré-estabelecido, o processo é repetido para que a quantidade de regras seja reduzida ainda mais. Todavia, se a quantidade de regras estiver dentro de um intervalo definido, a última fase será executada a partir do processamento do conjunto de regras na UP Principal (fase L) para obtenção das BRF finais (fase M).

É importante ressaltar que no presente trabalho, todo o conjunto de regras obtido a partir de um certo conjunto de dados será processado integralmente. No entanto, os exemplos usados para avaliar a acurácia e a complexidade das regras geradas não são integralmente usados nessa fase. Assim, um processo de validação cruzada *holdout* é aplicado dividindo o conjunto de exemplos em dados de treinamento e dados de teste. No presente projeto os dados foram divididos de forma estratificada em 70% para treino e 30% para teste. Desse modo, após a geração das BRF finais na fase M da Figura 4-1, essas mesmas BRF são reavaliadas de acordo com os exemplos da etapa de teste. Portanto, uma nova acurácia é calculada para cada uma das BRF obtidas na fase M.

Vale ressaltar que cada partição de cada um dos *chunks* é processada de maneira paralela e independente pelos executores. Isso significa que os conjuntos de BRF gerados por cada um dos executores são projetadas levando em consideração apenas uma fração dos dados. Esse fato torna os conjuntos de BRF gerados insuficientes para resolver o problema como um todo. Aliado a isso, ainda existe o fato de, em alguns casos, haver a presença de regras repetidas em algumas das BRF. Portanto, torna-se necessário que as BRF geradas por cada um dos executores sejam combinadas para chegar a um conjunto de BRF que não possua regras repetidas e que considere todo o conjunto de dados disponível.

No presente projeto, os valores da complexidade são calculados levando em consideração o número de regras e o número de antecedentes das regras contidas nas BRF obtidas na fase final do processamento. Assim, os valores de complexidade permanecem os mesmos obtidos na fase de treinamento.

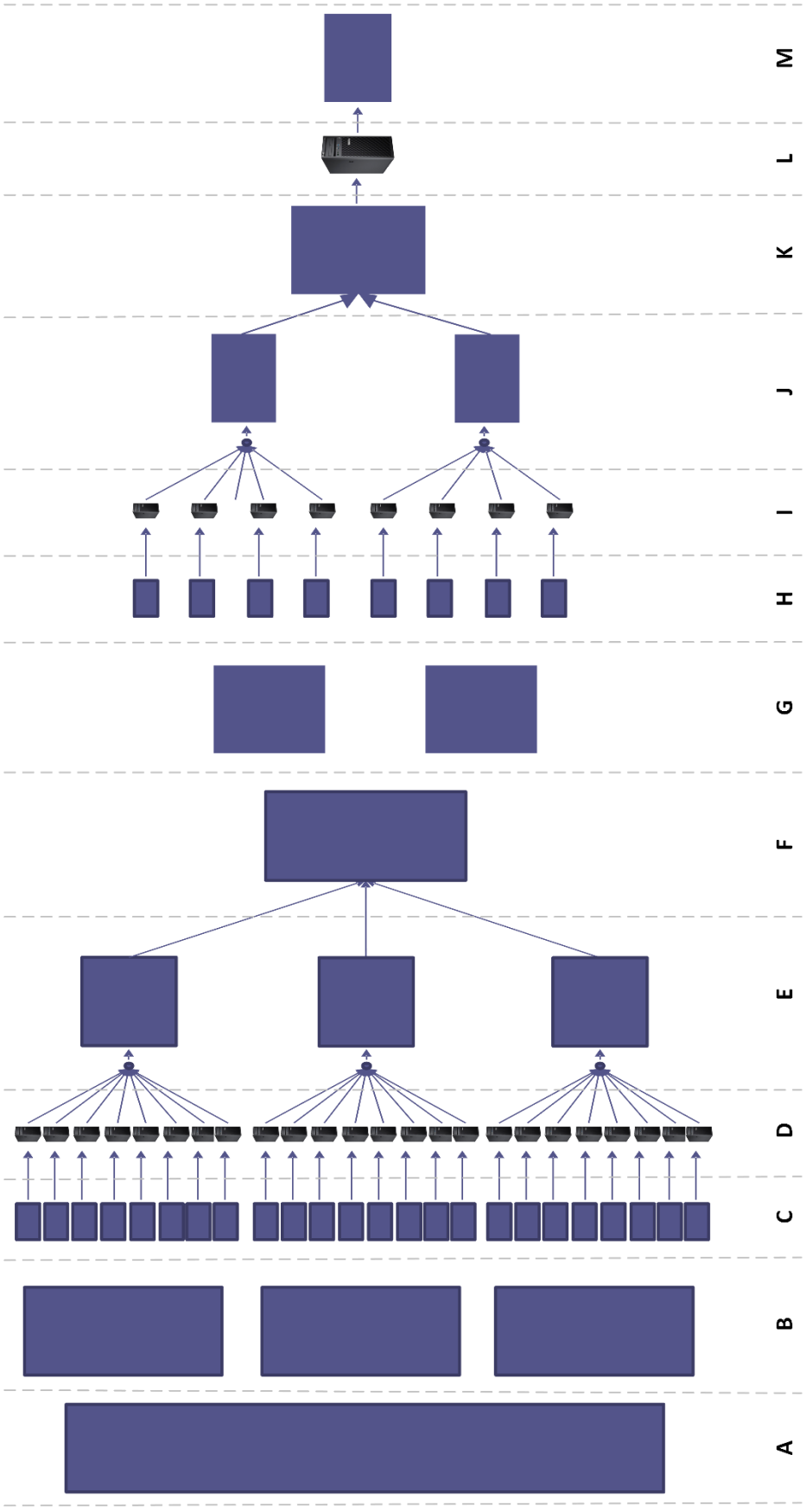

Figura 4-1 - Fases de processamento dos dados

4.4 Fluxograma do Algoritmo para Geração de BRF

Na Figura 4-2, tem-se o fluxograma do algoritmo desenvolvido para geração das BRF. Primeiramente o endereço do arquivo de regras é salvo. Em seguida, é definido o número que se pretende para cada uma das partições. O Número de Partições (NP) para o arquivo de regras é calculado pela relação entre o Número Total de Regras (NTR) a serem processadas e o Número de Regras por Partição (NRPP) que foi previamente definido.

$$NP = \frac{\text{NTR}}{\text{NRPP}} \qquad \text{[Eq. 4-1]}$$

Em seguida, o Número Máximo de Partições (NMP) é calculado. Como mencionado anteriormente, recomenda-se que esse valor seja de até 4 vezes o Número de Cores (NC) ou seja o número de núcleos de computação disponíveis no *cluster*.

$$NMP = 4 * NC \qquad \text{[Eq. 4-2]}$$

Com os valores de NP e NMP definidos, uma verificação é então realizada. O NP calculado de acordo com a quantidade de regras do arquivo de regras é maior que o Número Máximo de Partições (NMP)? Se sim, o arquivo de regras será dividido em *chunks* de acordo com o número de regras (NR), o NMP e o NRPP.

$$Número\ Chunks = \frac{\text{NR}}{\text{NMP} * \text{NRPP}} \qquad \text{[Eq. 4-3]}$$

Em seguida, o arquivo é dividido em *chunks* seguindo o valor obtido pela equação 4-3 e, para cada um dos *chunks*, o número de partições é então calculado seguindo a equação 4-1. Cada um dos *chunks* é então particionado e, de forma paralela, o SFG é aplicado para selecionar um subconjunto de regras de cada partição.

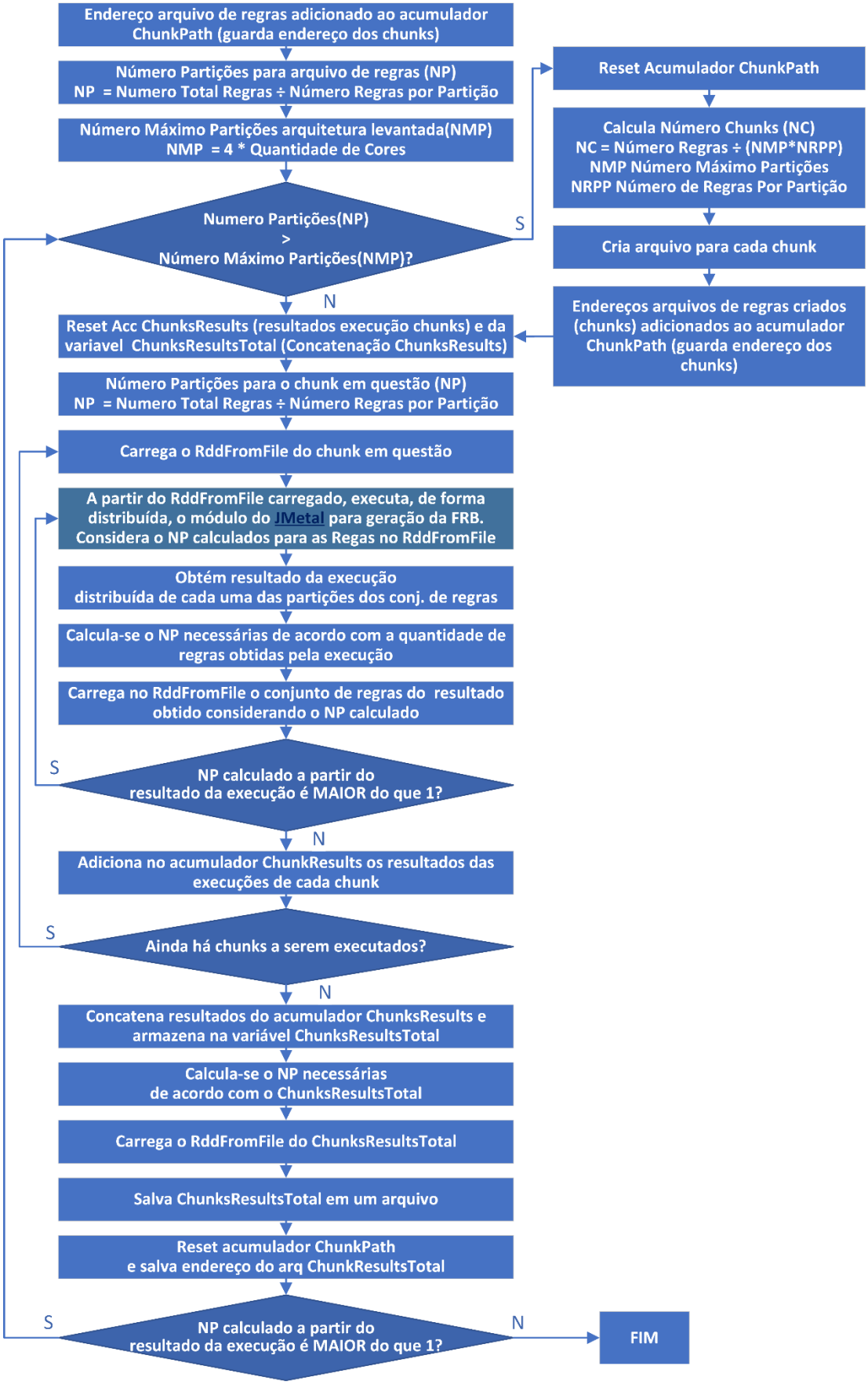

Figura 4-2 - Fluxograma algoritmo para geração das BRF

O módulo do SFG desenvolvido no *framework* jMetal é então executado em cada uma das UP e, assim, BRF são geradas. Cada um dos *chunks* são executados diversas vezes até que a quantidade de regras obtidas por aquele *chunk* esteja dentro de um valor pré-determinado. As BRF obtidas por cada um dos *chunks* são unidas em um grande grupo e é novamente verificada a quantidade de regras total das BRF geradas. Se a quantidade de regras total obtidas pelos *chunks* ainda for muito alta o processo é repetido até que a quantidade de regras esteja dentro de um intervalo pré-definido. Se a quantidade de regras estiver dentro de um valor pré-definido, as regras são processadas no nó principal e BRF finais são obtidas.

Esses valores pré-definidos variam para cada conjunto de dados e podem ser definidos de acordo com a quantidade de regras em cada partição. Se, por exemplo, para um determinado conjunto de dados uma partição contem 1000 regras, o critério de parada pode ser definido como 1000, ou seja, quando a soma de todas as regras obtidas em cada *chunk* for menor ou igual a 1000 regras, o critério de parada será atingido e essas 1000 ou menos regras formarão o conjunto de regras candidatas. Em seguida, uma última execução será realizada no computador central para extrair desse conjunto de regras candidatas as BRF finais.

4.5 Modelagem dos Indivíduos nos AGMO

Na estrutura desenvolvida, os indivíduos dos AGMO corresponderão às BRF. Sendo assim, o processo de otimização ocorre na busca por um conjunto de indivíduos com maior diversidade genética e que considere o balanceamento entre a acurácia e a complexidade das bases de regras.

Nesse sentido, os AGMO são usados, num primeiro momento, para a geração do conjunto de regras candidatas que pode ser visto na Figura 4-3. Esse conjunto de regras candidatas é obtido a partir das execuções em paralelo de cada nó do *cluster*. Com o conjunto de regras candidatas definido, ocorre uma última execução no nó principal e BRF finais são geradas.

Em relação aos indivíduos da população do AGMO, no método proposto, o número de genes do cromossomo é definido de acordo com o número de regras encontradas na base de regras produzidas pelo método FCA-*Based*. Para permitir a geração de bases de regras com um

número menor de regras, o método FCA-*Based* usa o valor [-1] para indicar que um gene representa uma regra inativa. O mesmo princípio foi usado no SFG desenvolvido nesse projeto. Portanto, cada gene do cromossomo pode assumir o valor [-1] indicando que a regra correspondente a aquele gene será inativada. Além disso, os genes poderão assumir valores correspondentes a cada uma das regras do arquivo de regras variando do valor [0] até a [quantidade de regras totais -1]. Cada valor desse intervalo corresponde a uma regra do arquivo de regras. Na Figura 4-3, *w* corresponde à última regra do conjunto de regras candidatas, *m* corresponde ao número de BRF geradas na execução do AGMO e *n* corresponde ao número de genes do cromossomo de cada indivíduo (equivalente à quantidade de atributos dos conjuntos de dados).

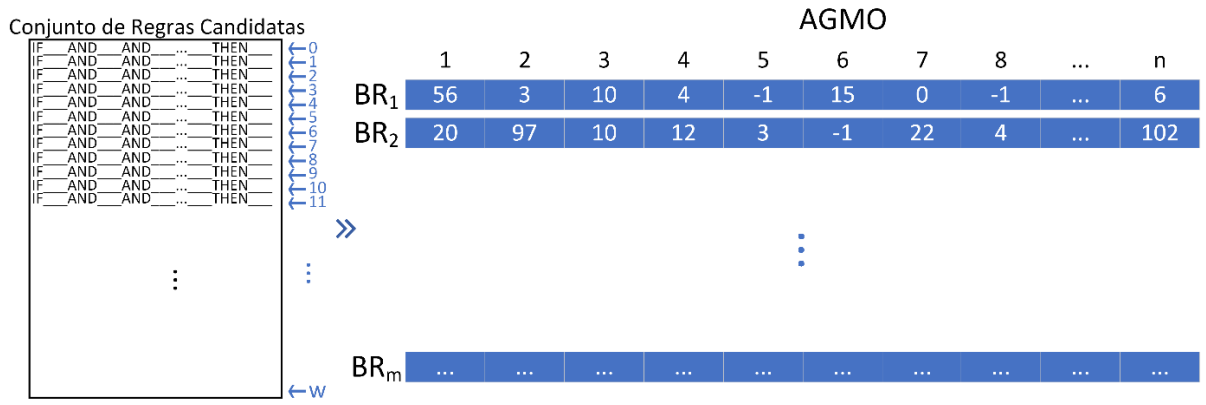

Figura 4-3 - Geração da BRF final a partir do conjunto de regras candidatas

Outro ponto importante é o da inicialização dos cromossomos. Baseada na estratégia proposta em Cintra, Camargo e Monard (2016), a inicialização dos cromossomos se dá seguindo as seguintes probabilidades:

- Probabilidade de 20% → 100% de regras ativas (nenhum gene assume valor [-1]);
- Probabilidade de 20% → 80% de regras ativas (20% dos genes assumem valor [-1]);
- Probabilidade de 20% → 60% de regras ativas (40% dos genes assumem valor [-1]);
- Probabilidade de 20% → 40% de regras ativas (60% dos genes assumem valor [-1]);
- Probabilidade de 20% → 20% de regras ativas (80% dos genes assumem valor [-1]);

Essas probabilidades indicam que, na criação da população inicial, cada indivíduo tem 20% de chance de assumir um desses casos. Por exemplo, se um indivíduo assumir o segundo caso, 80% dos genes desse indivíduo serão inicializados com valores correspondentes a regras

ativas e 20% dos genes serão inicializados com valores iguais a [-1], indicando a existência de regras inativas. Essa estratégia acelera a convergência e faz com que BRF mais compactas (com menos regras) sejam geradas o que, consequentemente, corrobora para que haja uma menor complexidade das BRF ao final do processo.

Em relação ao cálculo do *fitness, d*estaca-se que esses valores são calculados com vistas à redução do número de regras dos cromossomos (cada cromossomo corresponde a uma base de regras com quantidade de regras que pode variar de indivíduo a indivíduo) durante o processo de busca. O *fitness* é calculado com base na taxa de classificação correta e no número de regras e de antecedentes das regras.

4.6 Sistema *Fuzzy* Genético

No projeto foi realizada uma inclusão de um Sistema *Fuzzy* no *framework* jMetal. Foram incluídos métodos para a leitura dos arquivos de regras e de exemplos, métodos para a construção dos conjuntos *Fuzzy*, métodos para os cálculos das pertinências e métodos para os cálculos dos valores de acurácia e complexidade de cada uma das regras. Foram adicionadas estruturas responsáveis pelo número de Regras *Fuzzy*, pelo mapeamento da quantidade de antecedentes de cada regra *Fuzzy*, para receber os valores dos atributos, para a criação das partições *Fuzzy* e para a realização das inferências necessárias.

O sistema desenvolvido foi incorporado à estrutura do *framework* formando um SFG que foi usado para selecionar, a partir de um conjunto inicial de regras, aqueles que formariam as BRF.

4.6.1 Sistema *Fuzzy* Genético no *framework* jMetal

No projeto a ideia é usar o *framework* jMetal para o desenvolvimento do SFG que será usado para, a partir dos dados de entrada (conjunto de exemplos e de regras), extrair BRF. O jMetal é composto de vários módulos e alguns deles foram modificados para comportar o Sistema *Fuzzy* Genético proposto. É mostrado na Figura 4-4 um esquema de como o jMetal se comporta para o presente projeto.

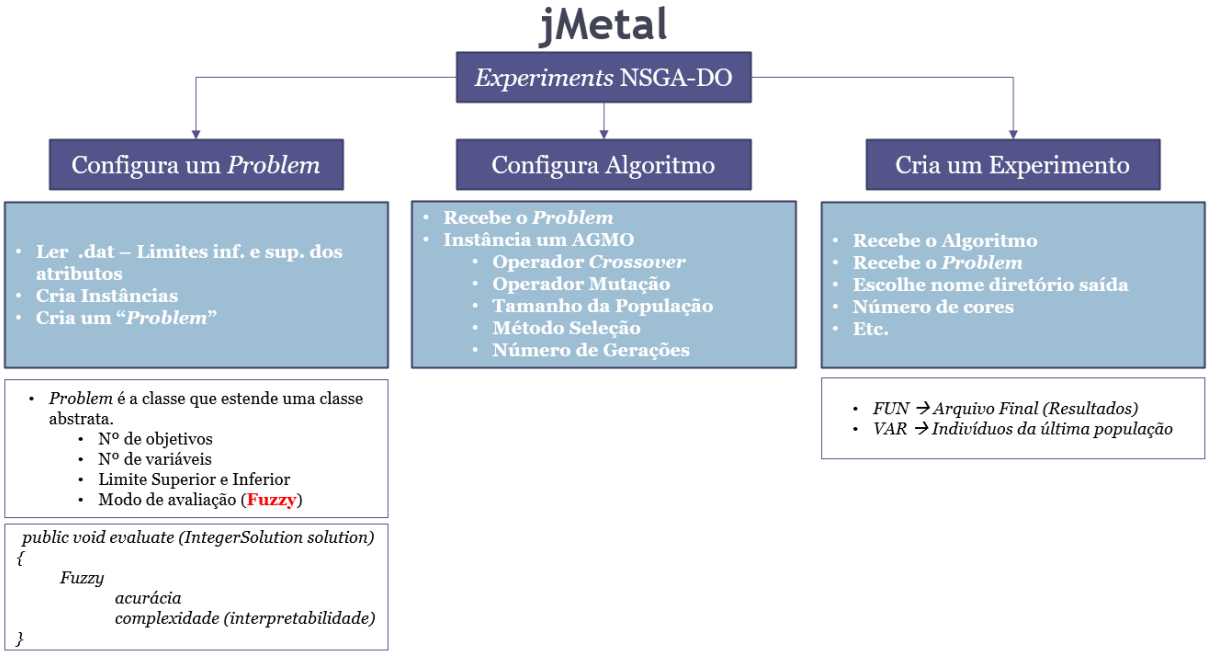

Figura 4-4 - Módulos do *framework* jMetal adaptados para criação do SFG

Na Figura 4-4 são apresentados os módulos de Configuração de Problema, Configuração de Algoritmo e configuração de Experimentos que foram adaptados para se adequar às necessidades do projeto.

No módulo *Configura Problem* foi desenvolvido o modulo *Fuzzy* onde são definidos os componentes do Sistema *Fuzzy* e onde cada indivíduo (BRF) do AG tem o seu *fitness* calculado (acurácia e complexidade). Foi usado o método de raciocínio *Fuzzy* Geral[19] (ganha a classe com maior grau de classificação) com os valores divididos em 3 conjuntos *Fuzzy* triangulares uniformemente distribuídos (baixo, médio e alto).

No módulo *Configura Algoritmo* os AGMO são configurados de acordo com o operador de *Crossover* (*IntegerSBXCrossover*), de Mutação (*IntegerPolynomialMutation*), tamanho da População (200 indivíduos), quantidade de gerações (1000 gerações) e Método de Seleção

[19] Método que agrega os graus de compatibilidade de todas regras e classifica o padrão de entrada usando a classe que possui maior grau de compatibilidade considerando todas regras de mesma classe (CINTRA, 2012).

(*BinaryTournamentSelection*)[20]. Os parâmetros escolhidos foram baseados no trabalho de Pimenta e Camargo (2015).

O módulo "*Cria Experimento*" recebe o *Problem* configurado e o AGMO instanciado e, assim, a execução é realizada gerando três arquivos de saída. Arquivo **FUN0** contendo os resultados de Acurácia e Complexidade de cada solução (BRF) de acordo com os exemplos para a fase de Teste. O arquivo FUN1 contendo os resultados de Acurácia e Complexidade de cada solução (BRF) de acordo com os exemplos para a fase de Teste e Treino e, por fim, o Arquivo VAR contendo o(s) cromossomo(s) da última população.

4.7 Integração dos Sistemas

Com o SFG desenvolvido, o próximo passo é a realização da integração entre os módulos do sistema. Na Figura 4-5 é apresentado um esquema dessa integração entre o SFG em Java, o software desenvolvido para possibilitar a paralelização dos processos (desenvolvido na linguagem Scala) e o sistema de serviço em nuvem *Amazon* EMR.

Após o desenvolvido do SFG, a ideia foi criar um arquivo JAR contendo o SFGMO e as Bases de Dados usadas para avaliação das BRF geradas pelos AGMO. O Arquivo de Regras obtido pelo método FCA_*Based* é então particionado e cada uma dessas partições (contendo várias regras) é distribuída para as *Workloaders* (Unidades de Processamento) juntamente com o arquivo JAR. Esse sistema é executado por meio do *Spark Shell*[21] da *Amazon* EMR para que o processamento seja realizado e BRF sejam geradas. O processo é repetido de acordo com as regras obtidas em cada interação até que, ao final do processo, um conjunto de BRF é obtido. O critério de parada consiste no número de regras geradas considerando todas as execuções paralelas de uma dada execução.

[20] Mais detalhes sobre os operadores podem ser encontrados na documentação do jMetal disponível em https://jmetal.readthedocs.io/en/latest/ .

[21] O *Spark Shell* permite criar programas na linguagem Scala e enviar trabalhos à estrutura de computadores distribuídos via conexão SSH e invocação, na linha de comando, da diretiva *spark-shell*.

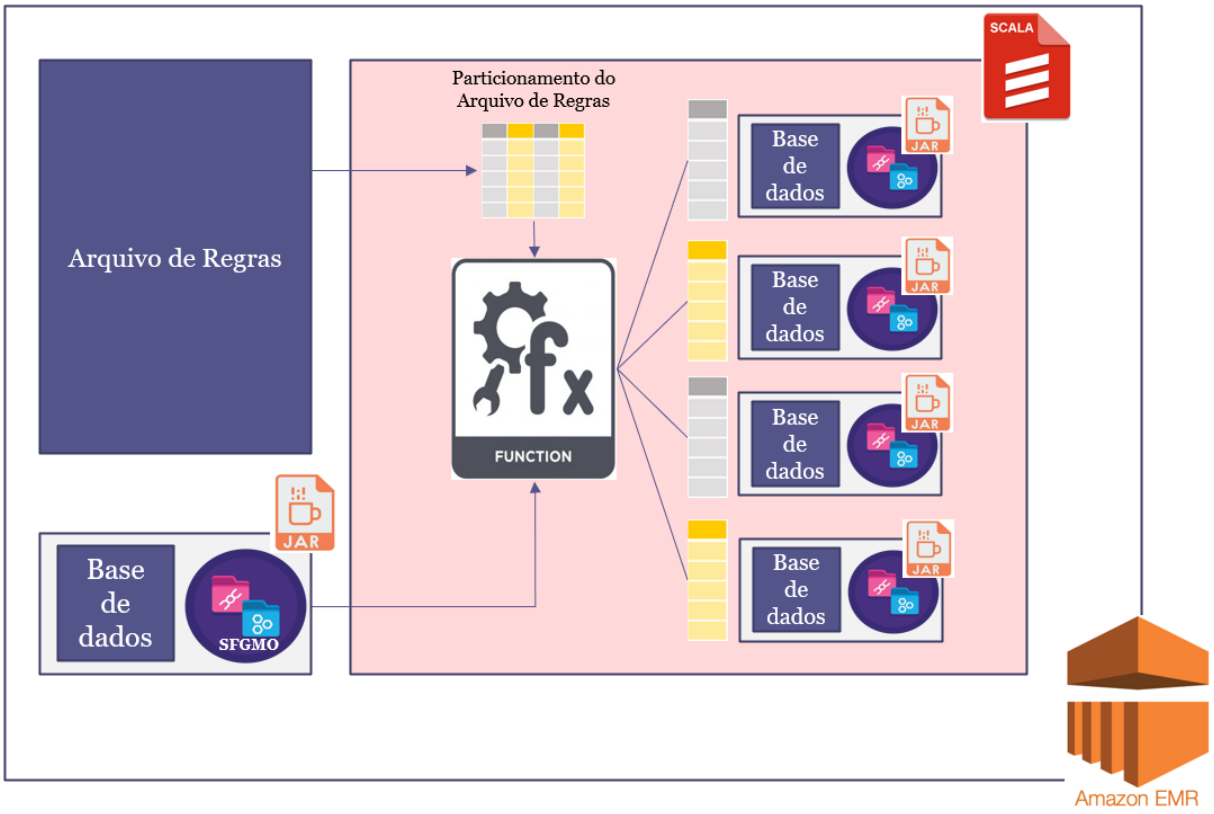

Figura 4-5 - Esquema do sistema integrado desenvolvido para extração de BRF com o uso da Computação Paralela.

4.8 Ferramentas

Principais ferramentas a serem usadas para a realização deste projeto são:

- Linguagem de Programação Scala
  - Escolhida por ser comumente usada pela comunidade cientifica na resolução de problemas que envolvem dados massivos. Também possui uma quantidade considerável de bibliotecas disponíveis para trabalhos que envolvem tratamento de dados e Computação Paralela.

- *Framework Apache Spark*
  - Escolhido pela velocidade que executa as soluções desenvolvidas para tratamento de dados no contexto do projeto. Executa operações em

memória, o que acelera o processamento. Possui a vantagem de ter seus componentes integrados na própria ferramenta além de permitir que a programação seja feita na linguagem Scala.

- *Framework* jMetal[22]
    - jMetal é um *framework* baseado em Java para otimização multiobjetivo usando metaheurísticas. É um pacote de software flexível, extensível e fácil de usar que tem sido usado em uma ampla variedade de aplicativos (DURILLO; NEBRO, 2011).

- *Amazon Web Service* (AWS)
    - Plataforma de computação em nuvem desenvolvida e disponibilizada pela empresa multinacional Amazon. Oferece serviços de computação como *Amazon* EC2, *Amazon* EMR, *Amazon* S3, *Apache Spark* e *Apache Yarn* (para gerenciamento do *cluster*) usados no desenvolvimento do trabalho.

- Ambiente de desenvolvimento Eclipse
    - Usado no desenvolvimento dos módulos incluídos no *framework* jMetal como os módulos do Sistema *Fuzzy* e de leitura de dados.

- Ambiente de desenvolvimento IntelliJ
    - Usado para desenvolver o código em Scala que foi executado na plataforma da AWS.

[22] Disponível em http://jmetal.sourceforge.net/, acessado em 15 de abril de 2022.

4.9 Considerações Finais

Nesse capítulo foi apresentada a proposta do trabalho abordado, as etapas da estratégia de modelagem desenvolvida e as principais ferramentas a serem usadas.

As plataformas de desenvolvimento e testes escolhidas para serem usadas no trabalho foram apresentadas e o algoritmo proposta para tratamento dos dados foi descrito. A seguir, no Capítulo 5, os testes serão apresentados e os resultados serão descritos e analisados.

Capítulo 5

EXPERIMENTOS E ANÁLISES DOS RESULTADOS

5.1 Considerações Iniciais

Para analisar o desempenho da proposta do trabalho foram realizados experimentos envolvendo alguns dos conjuntos de dados encontrados no repositório *UCI Repository of Machine Learning Databases*[23](LICHMAN, 2013), na plataforma aberta para compartilhamento de conjuntos de dados *OpenML*[24] e no KELL *Dataset Repository* de Alcalá-Fdez *et al.* (2011). Os experimentos foram realizados com os AGMO NSGA-DO, MNSGA-DO e NSGA-II. Todos eles foram executados seguindo as mesmas configurações.

Também foram realizados experimentos seguindo a abordagem serial (com execução em uma máquina local) e seguindo a abordagem paralela (com execução em um sistema distribuído). Além disso, foram realizados experimentos seguindo um método baseado na abordagem estado-da-arte FARC-HD onde foram usadas apenas regras com até 3 antecedentes.

Durante a execução dos experimentos foram considerados os seguintes aspectos:

- Dispersão das soluções na fronteira de Pareto
- Complexidade das BRF geradas
- Acurácia das BRF geradas
- Quantidade de soluções obtidas em cada execução
- Tempo de Execução

[23] Disponível em https://archive.ics.uci.edu/ml/index.php
[24] Disponível em https://www.openml.org/

5.2 Descrição dos Conjuntos de Dados

Na Tabela 5-1 são listados os conjuntos de dados usados nesse trabalho destacando as características de *Quantidade de Exemplos*, *Quantidade de Atributos*, *Quantidade de Classes* e *Quantidade de Regras* do arquivo FCA_Regras. Os domínios de cada um dos atributos estão representados por valores contínuos dentro de um intervalo. Esses valores foram usados para definir os conjuntos *Fuzzy*.

Tabela 5-1 - Conjuntos usados nos experimentos

	Conjunto de dados	Qntd. Exemplos	Qntd. Atributos	Qntd. Classes	Qntd. Regras
Bloco 1	*Appendicitis*	106	7	2	339
	Balance	625	4	3	1834
	Ecoli	336	7	8	532
	Glass	214	9	7	511
	Haberman	306	3	2	63
	Hayesroth	159	4	3	194
	Iris	150	4	3	61
	Monk-2	432	6	2	2772
	Newthyroid	215	5	3	75
	Pima	767	8	2	2438
	Saheart	462	9	2	7156
	Tae	151	5	3	275
	Titanic	2201	3	2	44
Bloco 2	*Cleveland*	297	13	5	36983
	Contraceptive	1473	9	3	18005
	Heart	270	13	2	38737
	Magic	19020	10	2	10659
	Towonorm	7400	20	2	1020457

Os conjuntos de dados foram divididos em dois blocos. O Bloco 1 é formado por conjuntos de dados cuja quantidade de regras obtidas pelo método FCA-*Based* é de até 10.000 regras e no máximo 10 variáveis em cada regra (9 antecedentes [atributos] e 1 consequente [saída da regra]). O Bloco 2 é formado por conjuntos de dados cujas regras correspondentes vão de cerca de 10.000 até cerca de 1 milhão de regras e variáveis que vão de 10 (9 antecedentes e 1 consequente) até 21 variáveis (20 antecedentes e 1 consequente).

A abordagem sequencial é testada e limitada apenas ao Bloco 1. Já a abordagem paralela é aplicada aos dois blocos de teste e os resultados são comparados e analisados.

5.3 Experimentos na AWS

Para realização do projeto o *Amazon Web Service* foi escolhido como plataforma para execução dos algoritmos. A escolha se deu devido às características adequadas aos testes necessários do projeto e ao suporte da *Amazon* que disponibilizou um ambiente distribuído de hardware, softwares e sistemas de armazenamento para que as aplicações pudessem ser executadas.

Na plataforma AWS, três serviços principais foram usados. O serviço *Amazon* EC2 que disponibilizou as instancias que foram usadas para execução do projeto. Foram usadas 8 instancias (1 nó principal + 7 nós secundários) do tipo **c5.xlarge** com cada uma delas contendo 4 CPUs, 8GB de memória RAM e largura de banda de 10Gbps. Foram usados, portanto, 32 núcleos para a execução dos testes.

Todas as instâncias c5.xlarge possuem Processadores *Intel Xeon Scalable* de segunda geração personalizados com uma frequência turbo sustentada de 3,6 GHz em todos os núcleos e uma frequência turbo de até 3,9 GHz em um único núcleo ou processador Intel Xeon Platinum série 8000 de primeira geração com uma frequência turbo sustentada de 3,4 GHz em todos os núcleos e uma frequência máxima turbo de até 3,5 GHz em um único núcleo. Essas instâncias C5 do Amazon EC2 são otimizadas para *workloads* com uso intensivo de computação e Inferência de *Machine Learning* e oferecem alta performance com bom custo-benefício por taxa de computação[25].

Outro serviço usado foi o *Amazon* EMR. Ele foi usado como base para a execução das aplicações permitindo a configurações de *clusters* com as instâncias executoras **c5.xlarge** do EC2 permitindo também a execução de um ambiente com *Apache Spark* onde aplicações em linguagens Scala e Java (usadas no trabalho) pudessem ser executadas.

O *Amazon* S3, serviço de armazenamento da *Amazon*, também foi usado. O arquivo de regras, resultados das execuções e os arquivos de log foram armazenados nesse repositório.

[25] Disponível em https://aws.amazon.com/pt/ec2/instance-types/, acessado em 15 de abril de 2022.

5.4 Experimentos Realizados

A Tabela 5-2 apresenta a configuração dos parâmetros dos AGMOs usados no trabalho. Os valores escolhidos foram baseados no trabalho de Pimenta e Camargo (2015). Para avaliação dos resultados foi usado o método de validação cruzada *holdout* com estratificação. Foram usados 70% dos exemplos para treinamento e 30% dos exemplos foram reservados para testes.

Tabela 5-2 - Parâmetros do AGMO

Configuração dos parâmetros dos AGMO	
Gerações	1000
Tamanho de População	200
Taxa de mutação	20%
Taxa de cruzamento	70%

Foram realizados 3 Grupos de testes como segue:

- Grupo A
  - Foram executados os testes para os conjuntos de dados do Bloco 1 (Tabela 5-1) numa abordagem sequencial que foi realizada em um computador Intel Core i5-11300H com 3.0GHz, 16GB RAM.
- Grupo B
  - Foram executados os testes para os conjuntos de dados do Bloco 1 e 2 numa abordagem paralela em um *cluster* da AWS com 8 instâncias (1 Principal e 7 *Workers*) cada uma contendo 4 CPU, 8GB de memória RAM e até 10Gbps de largura de banda de rede.
- Grupo C
  - Foram executados os testes para os conjuntos de dados do Bloco 1 e 2 numa abordagem paralela baseada na abordagem estado-da arte FARC-HD. Nesse teste, apenas regras com até 3 antecedentes foram usadas. O teste foi executado em um *cluster* da AWS com os mesmos tipos e quantidades de instâncias usadas nos testes do Grupo B.

Nas tabelas dos testes apresentadas a seguir, os algoritmos serão classificados seguindo o método de ranqueamento usado em Abbaszadeh e Hullermeier (2021). Assim, para cada linha

os algoritmos serão classificados em 1º, 2º ou 3º. Em seguida, é calculada a média das posições obtidas por cada algoritmo e quanto menor for o valor, melhor classificado será o algoritmo. Os testes serão apresentados a seguir.

5.5 Testes do Grupo A

Os testes do Grupo A foram realizados com os conjuntos de dados do Bloco 1 (Tabela 5-1) em uma abordagem sequencial. As taxas de Índice de Distribuição (ID) obtidas podem ser observadas na Tabela 5-3.

Na Tabela 5-3 a seta para baixo ↓ indica que, para os valores obtidos, quanto menor melhor. Em outras tabelas a serem apresentadas nesse capítulo, a seta para cima ↑ indicará o inverso, ou seja, para os valores obtidos, quanto maior for o resultado, melhor.

Tabela 5-3 - Índices de Distribuição (ID) das Soluções Centrais nas Fronteiras de Pareto (↓) – Grupo A

Dataset / Alg	NSGA-DO		MNSGA-DO		NSGA-II	
	ID ↓	*Ranking*	ID ↓	*Ranking*	ID ↓	*Ranking*
Appendicitis	0,4430	2	0,0000	1	0,5133	3
Balance	0,9387	3	0,6325	1	0,7948	2
Ecoli	1,0174	3	0,8044	1	0,9700	2
Glass	0,3572	1	0,8268	2	1,1205	3
Haberman	1,0299	2	0,4444	1	1,0426	3
Hayesroth	0,7975	1	1,0534	3	1,0166	2
Iris	0,2815	1	0,9901	3	0,5768	2
Monk-2	0,6492	2	0,7079	3	0,5695	1
Newthyroid	0,0000	1	0,6016	3	0,4674	2
Pima	0,6087	1	0,7669	2	0,7919	3
Saheart	0,7602	3	0,4958	1	0,7192	2
Tae	0,7061	3	0,4908	1	0,6996	2
Titanic	0,0000	1	0,0000	1	0,0000	1
Média (↓)		1,85		**1,77**		2,15
Ranking Algoritmo	2º		1º		3º	

Nos testes apresentados na Tabela 5-3, o algoritmo MNSGA-DO obteve a melhor média de ranqueamento indicando que ele apresenta uma melhor distribuição das soluções ao longo

da fronteira de Pareto. Como esse algoritmo busca não aglutinar mais de uma solução em um mesmo Ponto Ideal (vantagem em relação ao NSGA-DO) a tendência é que as soluções realmente fiquem mais distribuídas na fronteira de Pareto e, por consequência, o ID se aproxime mais do valor 0, que corresponderia a uma distribuição perfeita das soluções na fronteira de Pareto. A base Titanic indica ID igual a zero por possuir, para os 3 algoritmos, apenas uma BRF ao final das execuções. Apenas uma BRF também é obtida para a base Appendicitis na execução do MNSGA-DO.

Na Tabela 5-4 é apresentado o teste da Complexidade das soluções centrais obtidas com as execuções dos algoritmos.

Tabela 5-4 - Complexidade das Soluções Centrais nas Fronteiras de Pareto (↓) – Grupo A

Dataset / Alg	NSGA-DO		MNSGA-DO		NSGA-II	
	Complexidade ↓	*Ranking*	Complexidade ↓	*Ranking*	Complexidade ↓	*Ranking*
Appendicitis	0,1393	2	0,1452	3	0,1313	1
Balance	0,1650	3	0,1641	2	0,1614	1
Ecoli	0,1551	1	0,2129	3	0,2029	2
Glass	0,1487	2	0,1437	1	0,2297	3
Haberman	0,1455	1	0,1561	2	0,1587	3
Hayesroth	0,2043	2	0,2068	3	0,1965	1
Iris	0,1701	3	0,1578	1	0,1578	1
Monk-2	0,1536	2	0,1498	1	0,2073	3
Newthyroid	0,1760	1	0,1773	2	0,2027	3
Pima	0,1469	1	0,1509	3	0,1488	2
Saheart	0,1431	2	0,1435	3	0,1429	1
Tae	0,1836	3	0,1476	1	0,1531	2
Titanic	0,1250	3	0,0758	1	0,1212	2
Média (↓)		2,00		2,00		1,92
Ranking Algoritmo	2º		2º		1º	

Nos testes da Tabela 5-4 tem-se que o algoritmo NSGA-II obteve complexidade média menor em relação aos algoritmos NSGA-DO e MNSGA-DO, os quais obtiveram a mesma média para esse objetivo. Isso indica que o algoritmo evoluiu a ponto de obter BRF com um número menor de regras e de antecedentes nas regras. Entretanto, conforme apresentado na Tabela 5-5 a seguir, as acurácias correspondentes a essas BRF geradas pelo NSGA-II têm acurácia inferior às obtidas pelos algoritmos NSGA-DO e MNSGA-DO. Isso indica que, em detrimento da acurácia, o NSGA-II, privilegiou o objetivo da complexidade.

Na Tabela 5-5 o algoritmo MNSGA-DO obteve maior acurácia em média, indicando que além de uma maior distribuição das soluções na fronteira de Pareto o algoritmo conseguiu obter valores de acurácia melhores em relação ao NSGA-DO e ao NSGA-II. Isso indica que as soluções possuem uma taxa de classificação maior mesmo com a população final sendo mais heterogênea.

Tabela 5-5 - Acurácia das Soluções Centrais nas Fronteiras de Pareto (↑) – Grupo A

Dataset / Alg	NSGA-DO		MNSGA-DO		NSGA-II	
	Acurácia ↑	*Ranking*	Acurácia ↑	*Ranking*	Acurácia ↑	*Ranking*
Appendicitis	0,8438	1	0,8125	3	0,8438	1
Balance	0,7819	2	0,8032	1	0,7660	3
Ecoli	0,7030	3	0,7327	1	0,7327	1
Glass	0,5231	2	0,5385	1	0,4769	3
Haberman	0,7391	1	0,7391	1	0,7283	3
Hayesroth	0,6042	1	0,5417	2	0,0000	3
Iris	0,9111	3	0,9333	2	0,9778	1
Monk-2	0,5231	3	0,5538	1	0,5462	2
Newthyroid	0,8154	1	0,5692	3	0,6000	2
Pima	0,7489	2	0,6926	3	0,7749	1
Saheart	0,6763	3	0,7266	1	0,6978	2
Tae	0,5652	1	0,5217	2	0,4783	3
Titanic	0,0000	1	0,0000	1	0,0000	1
Média (↓)		1,85		1,69		2,00
Ranking Algoritmo	2º		1º		3º	

Na Tabela 5-6 percebe-se que, o NSGA-DO e o MNSGA-DO obtiveram quantidade média menor de soluções ao final das execuções em relação ao NSGAII. Isso indica que, como a complexidade média do NSGA-II foi menor, a quantidade de atributos das regras é menor contrabalanceando um maior número de regras obtidas ao final das execuções do NSGAII. Outro ponto importante é que, mesmo com um número menor de regras do que o NSGAII, o MNSGA-DO conseguiu obter acurácia em média maior, indicando que o algoritmo conseguiu evoluir para um menor número de soluções e, ainda assim, com taxa de classificação correta maior em relação aos outros algoritmos.

Tabela 5-6 - Quantidade de Soluções Finais (↓) – Grupo A

Dataset / Alg	NSGA-DO		MNSGA-DO		NSGA-II	
	Soluções ↓	*Ranking*	Soluções ↓	*Ranking*	Soluções ↓	*Ranking*
Appendicitis	4,00	2	2,00	1	4,00	2
Balance	11,00	2	13,00	3	10,00	1
Ecoli	8,00	1	10,00	2	12,00	3
Glass	5,00	1	5,00	1	9,00	3
Haberman	7,00	2	6,00	1	10,00	3
Hayesroth	5,00	1	9,00	2	12,00	3
Iris	8,00	2	8,00	2	4,00	1
Monk-2	7,00	2	7,00	2	5,00	1
Newthyroid	2,00	1	5,00	2	10,00	3
Pima	9,00	2	7,00	1	16,00	3
Saheart	5,00	1	7,00	2	7,00	2
Tae	6,00	3	4,00	1	5,00	2
Titanic	1,00	1	1,00	1	1,00	1
Média (↓)		1,62		1,62		2,15
Ranking Algoritmo	1º		1º		3º	

Na Tabela 5-7 são apresentados os tempos de execução de cada um dos algoritmos.

Tabela 5-7 - Tempo de Execução dos Algoritmos (↓) – Grupo A

Dataset / Alg	NSGA-DO			MNSGA-DO			NSGA-II		
	milisec.	minutos	*Ranking*	milisec.	minutos	*Ranking*	milisec.	minutos	*Ranking*
Appendicitis	5060	0,0843	2	4924	0,0821	1	5141	0,0857	3
Balance	146595	2,4433	1	148266	2,4711	2	152127	2,5355	3
Ecoli	17781	0,2964	2	18463	0,3077	3	16726	0,2788	1
Glass	13559	0,226	3	13531	0,2255	2	13193	0,2199	1
Haberman	584	0,0097	3	580	0,0097	2	539	0,009	1
Hayesroth	2070	0,0345	1	2098	0,035	2	2236	0,0373	3
Iris	405	0,0068	1	467	0,0078	3	423	0,0071	2
Monk-2	324452	5,4075	3	322828	5,3805	2	316842	5,2807	1
Newthyroid	559	0,0093	2	573	0,0096	3	538	0,009	1
Pima	362763	6,0461	2	344009	5,7335	1	368934	6,1489	3
Saheart	3070547	51,1758	3	2730156	45,5026	1	2845234	47,4206	2
Tae	3450	0,0575	3	3241	0,054	1	3400	0,0567	2
Titanic	1145	0,0191	1	1977	0,033	3	1155	0,0193	2
Média (↓)			2,08			2,00			1,92
Ranking Algoritmo	3º			2º			1º		

Percebe-se que o NSGA-II obteve menor tempo médio de execução. Nesse caso, como as operações de seleção do NSGA-II são menos complexas que as operações de seleção do NSGA-DO e MNSGA-DO, há o consumo menor dos recursos computacionais e, consequentemente, o tempo de execução passa a ser menor.

5.6 Testes do Grupo B

Os testes do Grupo B foram realizados com os dados do Bloco 1 e 2 (Tabela 5-1) em uma abordagem paralela. Um ponto importante a se destacar é que a quantidade de regras por partição na execução paralela pode variar. Testes preliminares foram realizados para verificar esses valores para todas as bases de dados. Os valores de Regras por Partição são apresentados na Tabela 5-8.

Tabela 5-8 - Quantidade de Regras por Partição

Dataset	Qntd de Regras por Partição
Appendicitis	100
Balance	500
Ecoli	600
Glass	100
Haberman	100
Hayesroth	200
Iris	100
Monk-2	500
Newthyroid	100
Pima	200
Saheart	100
Tae	100
Titanic	100
Cleveland	100
Contraceptive	1000
Heart	100
Magic	100
Towonorm	100

As taxas de ID obtidas podem ser observadas na Tabela 5-9.

Tabela 5-9 - ID das Soluções Centrais nas Fronteiras de Pareto (↓) – Grupo B

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		ID ↓	*Ranking*	ID ↓	*Ranking*	ID ↓	*Ranking*
Bloco 1	Appendicitis	0,4302	2	0,4291	1	0,5025	3
	Balance	1,0294	1	1,0714	2	1,2205	3
	Ecoli	1,1003	3	0,9384	1	1,0810	2
	Glass	0,5455	1	1,1693	3	0,7132	2
	Haberman	1,0785	3	0,7090	1	0,8270	2
	Hayesroth	0,8106	1	1,0093	3	0,9566	2
	Iris	0,8786	2	0,9201	3	0,6499	1
	Monk-2	0,8524	2	0,8183	1	0,9295	3
	Newthyroid	1,0283	3	0,5217	2	0,4560	1
	Pima	0,9998	2	0,6102	1	1,1749	3
	Saheart	0,9767	3	0,9471	2	0,5847	1
	Tae	0,5438	2	0,4964	1	0,6908	3
	Titanic	0,0000	1	0,0000	1	0,0000	1
Média Bloco 1(↓)			2,00		1,69		2,08
Ranking Algoritmo		2º		1º		3º	
Bloco 2	Cleveland	1,4086	3	0,6553	1	1,1356	2
	Contraceptive	0,5336	1	0,8768	2	1,0849	3
	Heart	1,2039	3	0,8791	1	1,0670	2
	Magic	1,0053	1	1,1786	3	1,1434	2
	Twonorm	1,2238	2	1,0267	1	1,2397	3
Média Bloco 2(↓)			2,00		1,6		2,4
Ranking Algoritmo		2º		1º		3º	
Média Geral (↓) Bloco 1 e Bloco 2			2,00		1,67		2,17
Ranking Algoritmo		2º		1º		3º	

Na Tabela 5-9, para o Bloco 1, o algoritmo MNSGA-DO obteve a melhor média de ranqueamento indicando que ele apresenta uma melhor distribuição das soluções ao longo da fronteira. O motivo ainda reside no fato do MNSGA-DO buscar não aglutinar mais de uma solução em um mesmo Ponto Ideal corroborando para que as soluções fiquem mais espalhadas na fronteira de Pareto mesmo em uma abordagem em Computação Paralela onde o arquivo de regras é particionado. Para o Bloco 2 que contém arquivos de regras maiores (variando de cerca de 10000 regras até mais de 1 milhão de regras), as questões relacionadas a como o MNSGA-

DO distribui as soluções, ainda persiste indicando que mesmo em uma abordagem onde os conjuntos de dados de entrada são maiores e o espaço de busca é dividido passando a ser explorado separadamente em cada um dos nós do *cluster*, o algoritmo ainda mantém a capacidade de espalhamento das soluções ao longo da fronteira. O NSGA-DO resultou em ID menor que o obtido para o NSGA-II indicando que as soluções ainda são obtidas de forma espalhada na fronteira (em relação ao NSGA-II), mas não tanto quanto o espalhamento alcançado pelo MNSGA-DO.

Na Tabela 5-10 tem-se os resultados para a Complexidade alcançada pelos algoritmos em um Contexto de Computação Paralela. Percebe-se que, para o Bloco 1, o MNSGA-DO alcançou resultados de complexidade menores. Já para o Bloco 2, com conjuntos de dados de entrada (conjunto de regras) maiores, o NSGA-II obteve resultados menores de complexidade indicando que, da mesma forma que ocorreu no ambiente sequencial, o NSGA-II priorizou o objetivo da complexidade em detrimento da acurácia (acurácia do NSGA-II foi menor que os demais algoritmos como pode ser visto no Bloco 2 da Tabela 5-10).

Ainda para a Tabela 5-10, considerando o Bloco 1 e o Bloco 2, a complexidade obtida pelo MNSGA-DO foi, em média, a menor. Como a acurácia do MNSGA-DO (ver Tabela 5-11 pág. 117) é maior que a dos demais algoritmos, percebe-se que o MNSGA-DO conseguiu reduzir a complexidade e ainda assim aumentou a acurácia das BRF obtidas.

Tabela 5-10 - Complexidade das Soluções Centrais nas Fronteiras de Pareto (↓) – Grupo B

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		Complexidade ↓	*Ranking*	Complexidade ↓	*Ranking*	Complexidade ↓	*Ranking*
Bloco 1	Appendicitis	0,1185	1	0,1487	3	0,1429	2
	Balance	0,2150	1	0,3088	2	0,3120	3
	Ecoli	0,1593	3	0,1570	2	0,1605	1
	Glass	0,2135	3	0,1593	1	0,1935	2
	Haberman	0,1792	3	0,1375	1	0,1732	2
	Hayesroth	0,1510	1	0,2926	3	0,2188	2
	Iris	0,1556	3	0,1494	2	0,1333	1
	Monk-2	0,2065	3	0,1400	1	0,1980	2
	Newthyroid	0,1981	2	0,1780	1	0,2333	3
	Pima	0,1524	1	0,1624	3	0,1603	2
	Saheart	0,1467	3	0,1323	1	0,1339	2
	Tae	0,1463	1	0,1654	2	0,1860	3
	Titanic	0,1667	3	0,1190	2	0,1146	1
Média Bloco 1(↓)			2,15		1,85		2,00
Ranking Algoritmo		3º		1º		2º	
Bloco 2	Cleveland	0,1916	3	0,1584	1	0,1745	2
	Contraceptive	0,1428	1	0,1853	2	0,3175	3
	Heart	0,1350	2	0,1371	3	0,1302	1
	Magic	0,1724	2	0,1835	3	0,1712	1
	Twonorm	0,1363	3	0,1317	1	0,1332	2
Média Bloco 2(↓)			2,20		2,00		1,80
Ranking Algoritmo		3º		2º		1º	
Média Geral (↓) Bloco 1 e Bloco 2			2,17		1,89		1,94
Ranking Algoritmo		3º		1º		2º	

Na Tabela 5-11 os resultados obtidos para a Acurácia são apresentados. Percebe-se que para o Bloco 1 e para o Bloco 2 o algoritmo MNSGA-DO obtive em média valores maiores de acurácia, indicando que, como dito anteriormente, o algoritmo alcançou, mesmo em um contexto distribuído, acurácias maiores com complexidades menores indicando que mesmo com menos regras ou com regras com menos antecedentes, as BRF obtidas possuem taxas de classificações corretas maiores.

Tabela 5-11 - Acurácia das Soluções Centrais nas Fronteiras de Pareto (↑) – Grupo B

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		Acurácia ↑	*Ranking*	Acurácia ↑	*Ranking*	Acurácia ↑	*Ranking*
Bloco 1	Appendicitis	0,7500	3	0,8438	1	0,8438	1
	Balance	0,6809	3	0,7553	1	0,6915	2
	Ecoli	0,7030	1	0,6832	2	0,6733	3
	Glass	0,5538	1	0,5231	2	0,0000	3
	Haberman	0,7283	2	0,0000	3	0,7609	1
	Hayesroth	0,5417	3	0,5833	2	0,7083	1
	Iris	0,9333	1	0,8222	2	0,8000	3
	Monk-2	0,6077	2	0,6308	1	0,5923	3
	Newthyroid	0,8462	2	0,8615	1	0,8462	2
	Pima	0,7446	1	0,7186	3	0,7403	2
	Saheart	0,6547	2	0,6619	1	0,6547	2
	Tae	0,5000	2	0,5000	2	0,5435	1
	Titanic	0,0000	1	0,0000	1	0,0000	1
Média Bloco 1(↓)			1,85		1,69		1,92
Ranking Algoritmo		2º		1º		3º	
Bloco 2	Cleveland	0,6333	3	0,6667	1	0,6667	1
	Contraceptive	0,4480	3	0,5000	1	0,5000	1
	Heart	0,8148	1	0,8148	1	0,7531	3
	Magic	0,7999	1	0,7545	2	0,7068	3
	Twonorm	0,9658	3	0,9680	1	0,9667	2
Média Bloco 2(↓)			2,20		1,20		2,00
Ranking Algoritmo		3º		1º		2º	
Média Geral (↓) Bloco 1 e Bloco 2			1,94		1,56		1,94
Ranking Algoritmo		2º		1º		2º	

Na Tabela 5-12, referente à quantidade de soluções finais, percebe-se que, para as bases do Bloco 1 os algoritmos NSGA-DO e MNSGA-DO obtiveram quantidades de regras equivalentes. Entretanto, para as bases do Bloco 2 o desempenho do MNSGA-DO foi superior indicando que o MNSGA-DO obteve desempenho melhor quando atuou em um contexto de Computação Paralela para a extração de BRF quando as bases de dados de entrada são maiores. Para bases maiores, a quantidade de soluções obtidas pelo NSGA-II foi menor que a quantidade obtida pelo NSGA-DO, indicando que, em um contexto de Computação Paralela para grande quantidade de dados, o NSGA-II se comporta melhor em relação ao NSGA-DO e em um contexto Paralelo com dados de entrada menores o NSGA-DO se comporta melhor que o NSGA-II.

Tabela 5-12 - Quantidade de Soluções Finais (↓) – Grupo B

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		Soluções ↓	*Ranking*	Soluções ↓	*Ranking*	Soluções ↓	*Ranking*
Bloco 1	Appendicitis	4,0000	1	6,0000	2	6,0000	2
	Balance	9,0000	1	9,0000	1	12,0000	3
	Ecoli	8,0000	1	10,0000	3	9,0000	2
	Glass	11,0000	3	5,0000	1	5,0000	1
	Haberman	10,0000	3	4,0000	1	8,0000	2
	Hayesroth	5,0000	1	9,0000	3	8,0000	2
	Iris	5,0000	1	7,0000	3	5,0000	1
	Monk-2	6,0000	1	6,0000	1	7,0000	3
	Newthyroid	10,0000	2	7,0000	1	11,0000	3
	Pima	8,0000	2	6,0000	1	8,0000	2
	Saheart	8,0000	2	8,0000	2	6,0000	1
	Tae	5,0000	2	4,0000	1	11,0000	3
	Titanic	1,0000	1	1,0000	1	1,0000	1
Média Bloco 1(↓)			1,62		1,62		2,00
Ranking Algoritmo		1º		1º		3º	
Bloco 2	Cleveland	9,0000	2	8,0000	1	9,0000	2
	Contraceptive	7,0000	1	7,0000	1	7,0000	1
	Heart	8,0000	2	6,0000	1	11,0000	3
	Magic	11,0000	3	8,0000	1	8,0000	1
	Twonorm	13,0000	3	7,0000	1	11,0000	2
Média Bloco 2(↓)			2,2		1,00		1,80
Ranking Algoritmo		3º		1º		2º	
Média Geral (↓) Bloco 1 e Bloco 2			1,78		1,44		1,94
Ranking Algoritmo		2º		1º		3º	

Tabela 5-13 – Tempo de Execução dos Algoritmos (↓) – Grupo B

Dataset / Alg	NSGA-DO			MNSGA-DO			NSGA-II		
	milisec.	minutos	*Ranking*	milisec.	minutos	*Ranking*	milisec.	minutos	*Ranking*
Appendicitis	3549	0,0592	1	3657	0,061	2	6922	0,1154	3
Balance	23984	0,3997	2	73435	1,2239	3	17028	0,2838	1
Ecoli	42833	0,7139	3	29037	0,484	1	30048	0,5008	2
Glass	7586	0,1264	3	7250	0,1208	2	6762	0,1127	1
Haberman	1833	0,0306	2	1726	0,0288	1	1934	0,0322	3
Hayesroth	1833	0,0306	1	6085	0,1014	2	6383	0,1064	3
Iris	1775	0,0296	3	1748	0,0291	2	1657	0,0276	1
Monk-2	39129	0,6522	2	58285	0,9714	3	31266	0,5211	1
Newthyroid	1884	0,0314	3	1780	0,0297	2	1777	0,0296	1
Pima	24786	0,4131	1	48377	0,8063	3	28143	0,4691	2
Saheart	69065	1,1511	3	57508	0,9585	1	63201	1,0534	2
Tae	5319	0,0887	2	4999	0,0833	1	5612	0,0935	3
Titanic	3349	0,0558	2	5535	0,0923	3	3320	0,0553	1
Média Bloco 1(↓)			2,15			2,00			1,85
Ranking Algoritmo		3º			2º			1º	
Cleveland	297217	4,9536	3	265008	4,4168	1	273538	4,559	2
Contraceptive	732671	12,2112	3	700082	11,668	1	700671	11,6779	2
Heart	561557	9,3593	1	700735	11,6789	2	702215	11,7036	3
Magic	861503	14,3584	2	893795	14,8966	3	816487	13,6081	1
Towonorm	33588132	559,8022	1	35360876	589,3479	3	34920877	582,0146	2
Média Bloco 2(↓)			2,00			2,00			2,00
Ranking Algoritmo		1º			1º			1º	
Média Geral (↓) Bloco 1 e Bloco 2			1,78			1,67			1,56
Ranking Algoritmo		3º			2º			1º	

Na Tabela 5-13 percebe-se que o NSGA-II obteve menor tempo em média de execução. No contexto de computação paralela, as operações de menor complexidade do NSGA-II corroboraram para que o tempo de execução seja menor. O NSGA-DO executou por mais tempo para o Bloco 1 e por tempo equivalente aos outros 2 algoritmos no contexto do Bloco 2. Isso indica um melhor desempenho quando as bases são maiores e um menor desempenho quando as bases são menores.

5.7 Testes do Grupo C

Os testes do Grupo C foram realizados com os dados do Bloco 1 e 2 (Tabela 5-1) em uma abordagem paralela, assim como ocorreu no Teste do Grupo B. A diferença para o Grupo B é que os testes do Grupo C seguem uma abordagem baseada no método proposto em Alcalá-Fdez, Alcalá e Herrera (2011). Nesse método é proposto que haja uma simplificação do conjunto de regras em que apenas regras com no máximo 3 antecedentes sejam usados. Uma pré-seleção de regras foi realizada e novos arquivos de regras foram criados contendo apenas as regras que possuem no máximo 3 antecedentes. Esses novos conjuntos de regras foram os usados para a realização dos testes do Grupo C.

A Tabela 5-14 apresenta as novas configurações dos conjuntos de regras usados.

Tabela 5-14 - Conjuntos usados nos experimentos e formados por regras com no máximo 3 antecedentes cada uma – Grupo C

	Conjunto de dados	Qntd. Exemplos	Qntd. Atributos	Qntd. Classes	Qntd. Regras
Bloco 1	*Appendicitis*	106	7	2	140
	Balance	625	4	3	1118
	Ecoli	336	7	8	67
	Glass	214	9	7	67
	Haberman	306	3	2	63
	Hayesroth	159	4	3	138
	Iris	150	4	3	41
	Monk-2	432	6	2	838
	Newthyroid	215	5	3	23
	Pima	767	8	2	423
	Saheart	462	9	2	1036
	Tae	151	5	3	149
	Titanic	2201	3	2	44
Bloco 2	*Cleveland*	297	13	5	2666
	Contraceptive	1473	9	3	2071
	Heart	270	13	2	2268
	Magic	19020	10	2	734
	Towonorm	7400	20	2	587

As taxas de ID, a Complexidade, a Taxa de Acurácia, A Quantidade de Soluções na fronteira e o Tempo de Execução do Grupo C podem ser observadas respectivamente na Tabela 5-15,

Tabela 5-16, Tabela 5-17, Tabela 5-18 e Tabela 5-19. Percebe-se que o NSGA-II obteve o melhor desempenho no Grupo C indicando que a simplificação dos conjuntos de regras acelera a convergência propiciando que um algoritmo com operações menos complexas obtenha melhores resultados que algoritmos mais complexos. Isso ocorre pelo fato da evolução do algoritmo ocorrer em um ambiente menos complexo fazendo com que o fato de os algoritmos NSGA-DO e MNSGA-DO possuírem operações mais complexas não se mostrar, nesse caso, como uma vantagem evolutiva.

Tabela 5-15 - Índices de Distribuição (ID) das Soluções Centrais nas Fronteiras de Pareto (↓) – Grupo C

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		ID ↓	*Ranking*	ID ↓	*Ranking*	ID ↓	*Ranking*
Bloco 1	Appendicitis	1,2337	3	0,3203	1	0,6279	2
	Balance	1,0663	3	0,6261	1	0,9466	2
	Ecoli	0,7486	2	0,9147	3	0,5107	1
	Glass	1,0692	3	0,9789	2	0,7024	1
	Haberman	1,2632	3	0,7896	2	0,7231	1
	Hayesroth	0,9470	2	0,7411	1	1,1683	3
	Iris	0,7503	1	1,4346	3	0,7743	2
	Monk-2	1,0658	2	1,1840	3	0,7067	1
	Newthyroid	0,2082	1	0,7244	3	0,6808	2
	Pima	0,4016	1	0,5267	2	0,6986	3
	Saheart	1,0250	3	0,2893	1	0,5234	2
	Tae	0,4251	1	0,7544	3	0,5919	2
	Titanic	0,0000	1	0,0000	1	0,0000	1
Média Bloco 1(↓)			2,00		2,00		1,77
Ranking Algoritmo		2º		2º		1º	
Bloco 2	Cleveland	0,8464	3	0,4503	1	0,4929	2
	Contraceptive	0,5442	2	0,4981	1	0,7147	3
	Heart	0,6761	1	0,7887	3	0,7081	2
	Magic	0,9281	3	0,7248	2	0,3636	1
	Twonorm	0,8580	1	1,1822	3	1,0076	2
Média Bloco 2(↓)			2,00		2,00		2,00
Ranking Algoritmo		1º		1º		1º	
Média Geral (↓) Bloco 1 e Bloco 2			2,00		2,00		1,83
Ranking Algoritmo		3º		2º		1º	

Tabela 5-16 - Complexidade das Soluções Centrais nas Fronteiras de Pareto (↓) – Grupo C

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		Complexidade ↓	*Ranking*	Complexidade ↓	*Ranking*	Complexidade ↓	*Ranking*
Bloco 1	Appendicitis	0,1236	2	0,1155	1	0,1281	3
	Balance	0,2072	3	0,1479	1	0,1518	2
	Ecoli	0,1822	2	0,2171	3	0,1589	1
	Glass	0,1629	2	0,1351	1	0,1764	3
	Haberman	0,1838	3	0,1622	2	0,1304	1
	Hayesroth	0,3110	3	0,1692	1	0,2794	2
	Iris	0,1613	2	0,1202	1	0,3750	3
	Monk-2	0,1990	1	0,2011	2	0,2016	3
	Newthyroid	0,2133	3	0,1917	1	0,2071	2
	Pima	0,1106	1	0,1583	2	0,1653	3
	Saheart	0,1213	2	0,1195	1	0,1298	3
	Tae	0,2355	3	0,1827	2	0,1312	1
	Titanic	0,1389	2	0,1389	2	0,0926	1
Média Bloco 1(↓)			2,23		1,54		2,15
Ranking Algoritmo		3º		1º		2º	
Bloco 2	Cleveland	0,1507	2	0,1077	1	0,1803	3
	Contraceptive	0,2116	3	0,1524	2	0,1140	1
	Heart	0,1106	2	0,1079	1	0,1108	3
	Magic	0,1171	1	0,1758	3	0,1368	2
	Twonorm	0,1033	1	0,1066	3	0,1049	2
Média Bloco 2(↓)			1,80		2,00		2,20
Ranking Algoritmo		1º		2º		3º	
Média Geral (↓) Bloco 1 e Bloco 2			2,11		1,67		2,17
Ranking Algoritmo		2º		1º		3º	

Tabela 5-17 - Acurácia das Soluções Centrais nas Fronteiras de Pareto (↑) – Grupo C

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		Acurácia ↑	*Ranking*	Acurácia ↑	*Ranking*	Acurácia ↑	*Ranking*
Bloco 1	Appendicitis	0,8438	1	0,8125	2	0,7813	3
	Balance	0,0000	2	0,0000	2	0,6489	1
	Ecoli	0,5545	2	0,5248	3	0,5842	1
	Glass	0,5231	1	0,4769	2	0,4769	2
	Haberman	0,7174	2	0,7609	1	0,0000	3
	Hayesroth	0,0000	3	0,4583	2	0,6667	1
	Iris	0,0000	3	0,8444	2	0,9556	1
	Monk-2	0,5615	3	0,7385	1	0,6538	2
	Newthyroid	0,7538	2	0,7538	2	0,7692	1
	Pima	0,7143	3	0,7316	2	0,7662	1
	Saheart	0,6835	1	0,6547	3	0,6835	1
	Tae	0,4565	3	0,5652	1	0,4783	2
	Titanic	0,0000	1	0,0000	1	0,0000	1
Média Bloco 1(↓)			2,08		1,85		1,54
Ranking Algoritmo		3º		2º		1º	
Bloco 2	Cleveland	0,5889	2	0,6000	1	0,5889	2
	Contraceptive	0,4774	2	0,4683	3	0,4887	1
	Heart	0,8272	1	0,7901	3	0,8148	2
	Magic	0,7404	1	0,7219	3	0,7313	2
	Twonorm	0,9324	1	0,9045	3	0,9230	2
Média Bloco 2(↓)			1,40		2,60		1,80
Ranking Algoritmo		1º		3º		2º	
Média Geral (↓) Bloco 1 e Bloco 2			1,89		2,06		1,61
Ranking Algoritmo		2º		3º		1º	

Tabela 5-18 - Quantidade de Soluções Finais (↓) – Grupo C

Dataset / Alg		NSGA-DO		MNSGA-DO		NSGA-II	
		Soluções ↓	*Ranking*	Soluções ↓	*Ranking*	Soluções ↓	*Ranking*
Bloco 1	Appendicitis	6,0000	1	7,0000	2	7,0000	2
	Balance	10,0000	2	10,0000	2	8,0000	1
	Ecoli	6,0000	2	6,0000	2	5,0000	1
	Glass	4,0000	1	7,0000	2	10,0000	3
	Haberman	7,0000	1	8,0000	3	7,0000	1
	Hayesroth	8,0000	1	9,0000	2	9,0000	2
	Iris	9,0000	2	9,0000	2	3,0000	1
	Monk-2	12,0000	3	7,0000	1	7,0000	1
	Newthyroid	6,0000	1	6,0000	1	10,0000	3
	Pima	8,0000	3	7,0000	1	7,0000	1
	Saheart	6,0000	2	5,0000	1	8,0000	3
	Tae	9,0000	2	9,0000	2	7,0000	1
	Titanic	1,0000	1	1,0000	1	1,0000	1
Média Bloco 1(↓)			1,69		1,69		1,62
Ranking Algoritmo		2º		2º		1º	
Bloco 2	Cleveland	6,0000	1	8,0000	3	7,0000	2
	Contraceptive	10,0000	3	7,0000	2	5,0000	1
	Heart	11,0000	3	9,0000	1	9,0000	1
	Magic	7,0000	2	6,0000	1	8,0000	3
	Twonorm	10,0000	1	17,0000	3	11,0000	2
Média Bloco 2(↓)			2,00		2,00		1,80
Ranking Algoritmo		2º		2º		1º	
Média Geral (↓) Bloco 1 e Bloco 2			1,78		1,78		1,67
Ranking Algoritmo		2º		2º		1º	

Tabela 5-19 - Tempo de Execução dos Algoritmos (↓) – Grupo C

Dataset / Alg	NSGA-DO			MNSGA-DO			NSGA-II		
	milisec.	minutos	*Ranking*	milisec.	minutos	*Ranking*	milisec.	minutos	*Ranking*
Appendicitis	690	0,0115	1	751	0,0125	2	1351	0,0225	3
Balance	9183	0,1531	2	28146	0,4691	3	6345	0,1058	1
Ecoli	1611	0,0269	3	1099	0,0183	1	1164	0,0194	2
Glass	310	0,0052	3	301	0,005	2	278	0,0046	1
Haberman	1741	0,029	3	1598	0,0266	2	810	0,0135	1
Hayesroth	1134	0,0189	1	3779	0,063	3	3770	0,0628	2
Iris	1424	0,0237	3	1366	0,0228	2	1198	0,02	1
Monk-2	3943	0,0657	2	6096	0,1016	3	3123	0,0521	1
Newthyroid	741	0,0124	2	714	0,0119	1	746	0,0124	3
Pima	852	0,0142	1	2136	0,0356	3	1081	0,018	2
Saheart	1094	0,0182	1	1182	0,0197	2	1281	0,0214	3
Tae	1853	0,0309	1	1874	0,0312	2	1990	0,0332	3
Titanic	3346	0,0558	1	5610	0,0935	3	3357	0,056	2
Média Bloco 1(↓)			1,85			2,23			1,92
Ranking Algoritmo		3º			2º			1º	
Cleveland	38136	0,6356	2	36548	0,6091	1	39845	0,6641	3
Contraceptive	27658	0,461	2	276722	4,612	3	23258	0,3876	1
Heart	32478	0,5413	2	38745	0,6458	3	31248	0,5208	1
Magic	18777	0,313	3	16589	0,2765	1	16873	0,2812	2
Towonorm	12567	0,2095	2	12587	0,2098	3	12489	0,2082	1
Média Bloco 2(↓)			2,20			2,20			1,60
Ranking Algoritmo		1º			1º			1º	
Média Geral (↓) Bloco 1 e Bloco 2			1,94			2,22			1,83
Ranking Algoritmo		2º			3º			1º	

5.8 Abordagem Sequencial e Abordagem Paralela

As métricas *Speedup* e *Efficiency* serão usadas no presente trabalho para medir o grau de desempenho dos algoritmos nos testes realizados. As tabelas Tabela 5-20, Tabela 5-21 e Tabela 5-22 apresentam o *Speedup* e a *Efficiency* para os Algoritmos NSGA-DO, MNSGA-DO

e NSGAII. As bases não presentes nas tabelas não foram executadas no modo sequencial devido à grande quantidade de regras de cada uma delas.

Tabela 5-20 - Cálculo do *Speedup* e da Eficiência para o Algoritmo NSGA-DO

Dataset	Sequencial (*milisec*)	Paralelo (*milisec*)	*Speedup*	*Efficiency*
Appendicitis	5060	3549	1,43	0,05
Balance	146595	23984	6,11	0,22
Ecoli	17781	42833	0,42	0,01
Glass	13559	7586	1,79	0,06
Haberman	584	1833	0,32	0,01
Hayesroth	2070	1833	1,13	0,04
Iris	405	1775	0,23	0,01
Monk-2	324452	39129	8,29	0,30
Newthyroid	559	1884	0,30	0,01
Pima	362763	24786	14,64	0,52
Saheart	3070547	69065	44,46	1,59
Tae	3450	5319	0,65	0,02
Titanic	1145	3349	0,34	0,01

Tabela 5-21 - Cálculo do *Speedup* e da Eficiência para o Algoritmo MNSGA-DO

Dataset	Sequencial (*milisec*)	Paralelo (*milisec*)	*Speedup*	*Efficiency*
Appendicitis	4924	3657	1,35	0,05
Balance	148266	73435	2,02	0,07
Ecoli	18463	29037	0,64	0,02
Glass	13531	7250	1,87	0,07
Haberman	580	1726	0,34	0,01
Hayesroth	2098	6085	0,34	0,01
Iris	467	1748	0,27	0,01
Monk-2	322828	58285	5,54	0,20
Newthyroid	573	1780	0,32	0,01
Pima	344009	48377	7,11	0,25
Saheart	2730156	57508	47,47	1,70
Tae	3241	4999	0,65	0,02
Titanic	1977	5535	0,36	0,01

Tabela 5-22 - Cálculo do *Speedup* e da Eficiência para o Algoritmo NSGAII

Dataset	Sequencial (*milisec*)	Paralelo (*milisec*)	*Speedup*	*Efficiency*
Appendicitis	5141	6922	0,74	0,03
Balance	152127	17028	8,93	0,32
Ecoli	16726	30048	0,56	0,02
Glass	13193	6762	1,95	0,07
Haberman	539	1934	0,28	0,01
Hayesroth	2236	6383	0,35	0,01
Iris	423	1657	0,26	0,01
Monk-2	316842	31266	10,13	0,36
Newthyroid	538	1777	0,30	0,01
Pima	368934	28143	13,11	0,47
Saheart	2845234	63201	45,02	1,61
Tae	3400	5612	0,61	0,02
Titanic	1155	3320	0,35	0,01

Nota-se que, para diversas bases, o *Speedup* resultou em um valor menor que 1. Isso indica que o *overhead* de ativação do *cluster* não compensou em relação ao tempo de execução para bases menores. Todavia, para bases maiores como a Pima (2.438 regras), Monk-2 (2.772 regras) e a Saheart (7.157 regras) o valor do *Speedup* foi maior do que 1 indicando que o tempo de execução do sistema de forma paralela foi realizado em menos tempo do que a execução seguindo a abordagem sequencial. Em relação a eficiência, para os 3 algoritmos, apenas a base Saheart obteve *Efficiency* acima de 0,5 indicando que, para essa base, o uso dos processadores está mais otimizado (mais computadores sendo usados por mais tempo). Em contrapartida, como o valor de *Efficiency* foi menor que 0,5 para todas as outras bases, isso indica que o *cluster* está sendo subutilizado sendo recomendando que bases maiores sejam usadas para fazer jus ao *overhead* de ativação do *cluster*.

5.9 Abordagem proposta e a abordagem baseada no método FARC-HD

Com o intuito de analisar a significância estatística das diferenças entre a abordagem proposta no trabalho (que considera todo o conjunto de regras) e a abordagem baseada no método FARC-HD (que considera apenas as regras com até 3 antecedentes) foi aplicado o teste de *Wilcoxon*.

Tabela 5-23 - Teste de Wilcoxon

Abordagem Proposta vs Abordagem Baseada no FARC-HD	R+	R-	H0 (a = 0,05)	Valor p
Distribuição ↓	926	400	Rejeitada	0.0139
Complexidade ↓	999	486	Rejeitada	0.0275
Acurácia ↑	960	366	Rejeitada	0.0055
Qntd. Soluções ↓	502	488	Aceita	0.9399
Tempo Execução ↓	1480	5	Rejeitada	0.0001

Supondo como hipótese nula *h0* que *não há diferença significativa entre as soluções obtidas pela Abordagem Proposta e as soluções obtidas pela Abordagem Baseada no método FARC-HD*, pode-se observar que a Abordagem Proposta se mostrou estatisticamente mais significativa no que diz respeito à acurácia obtida que a abordagem baseada no método FARC-HD.

Em contrapartida o valor da dispersão da abordagem proposta foi maior devido ao fato da abordagem baseada no FARC-HD lidar com menos regras em um mesmo número de gerações fazendo com que os algoritmos convirjam mais rápido e encontrem soluções com menor índice de dispersão. O mesmo acontece em relação à complexidade. Por lidar com menos regras a abordagem baseada no FARC-HD converge, no mesmo número de gerações, para soluções com menor número de regras e de antecedentes das regras. Todavia, por não explorar todo o espaço de busca, as soluções encontradas pelo método baseado no FARC-HD possuem estatisticamente menor significância em relação à acurácia do que a abordagem Proposta no trabalho.

Outro ponto interessante é que a quantidade de soluções da BRF final não foi significativamente diferente entre as abordagens. Isso demonstra que a abordagem Proposta no presente trabalho alcançou níveis maiores de acurácia para uma quantidade equivalente de soluções na BRF final (em relação à abordagem baseada no FARC-HD). Já o tempo de execução, como esperado, foi significativamente diferente. Isso ocorre pelo fato do método baseado em FARC-HD não explorar todo o espaço de busca de regras o que faz com que a execução ocorra em menor tempo.

5.10 Considerações Finais

Neste capítulo foram apresentados os experimentos realizados com os algoritmos NSGADO, MNSGA-DO e NSGAII. Esses algoritmos foram aplicados na seleção de regras *Fuzzy* a partir de um conjunto de regras gerados pelo método FCA-*Based.*

Os experimentos demonstraram que a abordagem proposta foi capaz de executar de forma distribuídas as bases de dados obtendo resultados de dispersão, complexidade e acurácia mesmo com os conjuntos de regras sendo particionados e processados separadamente de forma distribuída.

A partir dos resultados apresentados, no próximo capítulo será realizada a conclusão da tese levantando os pontos de contribuição e as limitações da abordagem desenvolvida. Além disso serão apresentadas as possibilidades de trabalhos futuros.

Capítulo 6

CONCLUSÃO

6.1 Considerações Iniciais

Neste trabalho, foi apresentada a proposta de uma estratégia de modelagem para a geração de Bases de Regras *Fuzzy* em problemas que envolvem dados com grande volume e alta dimensionalidade.

Neste capítulo será realizada a conclusão do trabalho abrangendo as contribuições, limitações e os trabalhos futuros.

6.2 Contribuições

Uma das principais contribuições desse trabalho está na ideia da criação de um modelo a ser usado no processo de extração de BRF a partir de grandes conjuntos de regras. Esse processo se dá de forma distribuída com o intuito de que sejam extraídas regras ponderadas entre os objetivos de acurácia e complexidade.

Testes foram realizados com arquivos de regras, primeiramente, contendo menos de 10000 regras e menos de 10 variáveis e, em um segundo momento, arquivos variando entre cerca de 10000 até cerca de 1 milhão de regras e com quantidade de variáveis abrangendo de 10 até 21 variáveis. Os resultados demonstraram a viabilidade da abordagem proposta uma vez que foi possível a obtenção de BRF acuradas de grandes conjuntos de dados por meio do uso da Computação Paralela.

Assim cumpriu-se o objetivo do trabalho que foi o de:

Desenvolver uma abordagem que faz uso da Computação Paralela para geração de Bases de Regras Fuzzy em problemas que envolvem grande volume e alta dimensionalidade dos dados usando os Algoritmos Genéticos Multiobjetivo de Ordenação por não-Dominância NSGA-DO proposto em Pimenta e Camargo (2015) e MNSGA-DO proposto em Machado et al. (2021) e NSGAII de Deb et al., (2002).

Outras contribuições foram relacionadas à investigação da aplicação dos AGMO NSGA-DO, MNSGA-DO e NSGA-II no contexto de Computação Paralela para extração de BRF a partir de conjuntos de regras obtidos pelo método FCA-*Based*, investigação do comportamento do AGMO MNSGA-DO no contexto de SFBR e a investigação do poder de representação e de compactação das BRF obtidas a partir dos conjuntos de regras extraídas de conjuntos de dados pelo método FCA-*Based*.

Além disso houve contribuições na direção de pesquisas voltadas ao uso da Computação Paralela na área de SFEMO e na direção de um modelo para aplicação dos diversos AGMO presentes no *framework* de código aberto jMetal em ambientes distribuídos. Outra contribuição foi em relação ao desenvolvimento de um Sistema *Fuzzy* no *framework* de código aberto jMetal, possibilitando que esse possa ser aplicado em ambientes distribuídos.

6.3 Limitações e Trabalhos Futuros

A partir deste trabalho surgem novas oportunidades de pesquisa, tanto no sentido de melhorar a abordagem proposta, quanto na elaboração de novas ideias de como usar a Computação Paralela no contexto de SFEMO.

Como limitação, a abordagem proposta não gera as regras a partir das bases de dados. Essas regras são obtidas a partir de um método de extração chamado de FCA-*Based*. Com as regras obtidas pelo método mencionado, uma análise é feita nos arquivos de regras para verificar, nomes das variáveis, quantidade de variáveis, quantidade de regras, etc. Só então é

que o arquivo de regras pode ser usado pelo sistema para geração de BRF. Outro ponto é que, devido, as limitações de poder de processamento e devido ao tamanho de algumas bases de regas (milhares e até milhões de regras) a abordagem sequencial é testada apenas para parte dos conjuntos de regras.

Como trabalho futuro, tem-se a ideia do desenvolvimento de AGMO em uma linguagem que permita executar os algoritmos em sistemas distribuídos como, por exemplo a linguagem Scala. Isso seria interessante pois poderia trazer benefícios no que diz respeito à paralelização dos processos internos do AG como operações de *crossover* e principalmente do cálculo do *fitness.* Abordagens que visam a paralelização dos processos de análises dos conjuntos de exemplos também poderiam trazer ganhos significativos ao processo uma vez que essa é uma das etapas mais demoradas da execução do modelo proposto.

REFERÊNCIAS

ABBASZADEH, Sadegh; HULLERMEIER, Eyke. Machine Learning with the Sugeno Integral: The Case of Binary Classification. **IEEE Transactions on Fuzzy Systems**, [*s. l.*], v. 29, n. 12, p. 3723–3733, 2021. Disponível em: https://doi.org/10.1109/TFUZZ.2020.3026144

ABRAHAM, Ajith; JAIN, Lakhmi; GOLDBERG, Robert. **Evolutionary Multiobjective Optimization: Theoretical Advances and Applications**. [*S. l.*]: Springer, 2004.

AGHAEIPOOR, Fatemeh; JAVIDI, Mohammad Masoud; FERNANDEZ, Alberto. IFC-BD: An Interpretable Fuzzy Classifier for Boosting Explainable Artificial Intelligence in Big Data. **IEEE Transactions on Fuzzy Systems**, [*s. l.*], v. 30, n. 3, p. 830–840, 2022. Disponível em: https://doi.org/10.1109/TFUZZ.2021.3049911

AGRAWAL, Rakesh; SRIKANT, Ramakrishnan. Fast Algorithms for Mining Association Rules in Large Databases. *In*: , 1994, San Francisco-CA-USA. **Proc. of the 20th International Conference on Very Large Data Bases (VLDB'94)**. San Francisco-CA-USA: Morgan Kaufmann Publishers Inc., 1994. p. 487–499.

ALCALÁ-FDEZ, J *et al.* KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. [*s. l.*], v. 17, p. 255–287, 2011. Disponível em: http://the-data-mine.com/bin/view/Software. Acesso em: 15 jul. 2022.

ALCALÁ-FDEZ, Jesús; ALCALÁ, Rafael; HERRERA, Francisco. A Fuzzy Association Rule-Based Classification Model for High-Dimensional Problems With Genetic Rule Selection and Lateral Tuning. **IEEE Transactions on Fuzzy Systems**, [*s. l.*], v. 19, n. 5, p. 857–872, 2011.

ALMASI, George S.; GOTTLIEB, Allan. **Highly Parallel Computing**. Redwood City - CA: Benjamin/Cummings Pub. Co, 1994. *E-book*.

AMDAHL, Gene M. Validity of the single processor approach to achieving large scale computing capabilities. *In*: , 1967. **AFIPS Conference Proceedings - 1967 Spring Joint Computer Conference, AFIPS 1967**. [*S. l.: s. n.*], 1967. Disponível em: https://doi.org/10.1145/1465482.1465560

ARAUJO, Antonio; KROHLING, Renato A. Generating a fuzzy rule based classification system by genetic learning of granularity level using TOPSIS. *In*: , 2019, Prague - República Tcheca. (Antonio E Araujo & Renato A Krohling, Org.)**Proceedings of the 11th Conference of the European Society for Fuzzy Logic and Technology, EUSFLAT**. Prague - República Tcheca: [*s. n.*], 2019. p. 482–489.

ÁVILA, Sergio Luciano. **Otimização Multiobjetivo e Análise de Sensibilidade para Concepção de Dispositivos**. 159 f. 2006. - Universidade Federal de Santa Catarina, [*s. l.*], 2006. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/88982/224404.pdf?sequence=1&isAllowed=y

BHUKYA, Hanumanthu; SADANANDAM, Manchala. Handling uncertainty using optimal clustering with rough sets-based rule generation model for data classification. **Expert Systems**, [*s. l.*], p. e13026, 2022. Disponível em: https://doi.org/10.1111/EXSY.13026. Acesso em: 4 jul. 2022.

CÁRDENAS, Edward Hinojosa; CAMARGO, Heloisa de Arruda. Multi-Objective Iterative Genetic Approach for Learning Fuzzy Classification Rules with Semantic-Based Selection of the Best Rule. *In*: , 2013, Edmonton - Canada. **2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)**. Edmonton - Canada: IEEE, 2013. p. 292–297. Disponível em: https://doi.org/10.1109/IFSA-NAFIPS.2013.6608415. Acesso em: 22 mar. 2017.

CÁRDENAS, Edward Hinojosa; CAMARGO, Heloisa de Arruda. Multiobjective Genetic Generation of Fuzzy Classifiers Using the Iterative Rule Learning. *In*: , 2012, Brisbane - Australia. **2012 IEEE International Conference on Fuzzy Systems**. Brisbane - Australia: IEEE, 2012. p. 1–8. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2012.6251183. Acesso em: 22 mar. 2017.

CÁRDENAS, Edward Hinojosa; CAMARGO, Heloisa de Arruda; TUPAC, V. Yvan J. Learning Fuzzy Classification Rules from Imbalanced Datasets using Multi-Objective Evolutionary Algorithm. *In*: , 2015, Curitiba - Brazil. **2015 Latin America Congress on Computational Intelligence (LA-CCI)**. Curitiba - Brazil: IEEE, 2015. p. 1–6. Disponível em: https://doi.org/10.1109/LA-CCI.2015.7435959. Acesso em: 22 mar. 2017.

CAREY, Scott. **Hadoop vs Spark: Which is right for your business?**. [*S. l.*], 2017. Disponível em: https://www.computerworld.com/article/3427163/hadoop-vs-spark--which-is-right-for-your-business--pros-and-cons--vendors--customers-and-use-cases.html. Acesso em: 20 maio 2019.

CASTRO, P. A. D.; CAMARGO, H. A. Learning and optimization of fuzzy rule base by means of self-adaptive genetic algorithm. **2004 IEEE International Conference on Fuzzy Systems**, Budapest - Hungary, v. 2, p. 1037–1042, 2004. Disponível em: https://doi.org/10.1109/FUZZY.2004.1375552

CHEN, Min; MAO, Shiwen; LIU, Yunhao. Big data: A survey. *In*: , 2014. **Mobile Networks and Applications**. [*S. l.: s. n.*], 2014. Disponível em: https://doi.org/10.1007/s11036-013-0489-0

CHI, Zheru; YAN, Hong; PHAM, Tuan. **Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition**. Singapore: WORLD SCIENTIFIC, 1996. (Advances in Fuzzy Systems — Applications and Theory).v. 10 Disponível em: https://doi.org/10.1142/3132. Acesso em: 1 out. 2019.

CINTRA, M E; CAMARGO, H A. Geração de Regras Fuzzy com Pré-Seleção de Regras Candidatas. **VI Encontro Nacional de Inteligência Artificial–ENIA**, Rio de Janeiro - Brazil, v. 1, p. 1341–1350, 2007. Disponível em: http://www.bdtd.ufscar.br/htdocs/tedeSimplificado/tde_busca/processaArquivo.php?codArquivo=1582. Acesso em: 29 ago. 2014.

CINTRA, Marcos E.; CAMARGO, Heloisa de Arruda; MONARD, Maria C. Genetic generation of fuzzy systems with rule extraction using formal concept analysis. **Information Sciences**, [*s. l.*], v. 349–350, p. 199–215, 2016. Disponível em: https://doi.org/10.1016/j.ins.2016.02.026. Acesso em: 22 mar. 2017.

CINTRA, Marcos E.; MONARD, Maria C.; CAMARGO, Heloisa de Arruda. Using fuzzy formal concepts in the genetic generation of fuzzy systems. *In*: , 2012, Brisbane - Australia. **2012 IEEE International Conference on Fuzzy Systems**. Brisbane - Australia: IEEE, 2012. p. 1–8. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2012.6251310. Acesso em: 22 mar. 2017.

CINTRA, Marcos Evandro. **Genetic Generation of Fuzzy Knowledge Bases: New**

Perspectives. 179 f. 2012. - University of Sao Paulo - USP, [*s. l.*], 2012.

COELLO, Carlos A Coello; LAMONT, Gary B; VELDHUIZEN, David A Van. **Evolutionary algorithms for solving multi-objective problems**. New York - NY - USA: Springer Science, Business Media, 2007. *E-book*.

DARWIN, Charles. **On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life**. London - England: London : John Murray, Albemarle Street, 1859. Disponível em: https://doi.org/10.1126/science.146.3640.51-b

DEAN, Jeffrey; GHEMAWAT, Sanjay. MapReduce: Simplified data processing on large clusters. **Communications of the ACM**, [*s. l.*], 2008. Disponível em: https://doi.org/10.1145/1327452.1327492

DEB, K. *et al.* A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, [*s. l.*], v. 6, n. 2, p. 182–197, 2002. Disponível em: https://doi.org/10.1109/4235.996017. Acesso em: 7 fev. 2019.

DEB, Kalyanmoy. Multi-Objective Evolutionary Algorithms. *In*: SPRINGER HANDBOOK OF COMPUTATIONAL INTELLIGENCE. Berlin, Heidelberg: Springer, 2015. p. 995–1015. Disponível em: https://doi.org/10.1007/978-3-662-43505-2_49

DUCANGE, Pietro; FAZZOLARI, Michela; MARCELLONI, Francesco. An overview of recent distributed algorithms for learning fuzzy models in Big Data classification. **Journal of Big Data**, [*s. l.*], v. 7, n. 19, p. 1–29, 2020. Disponível em: https://doi.org/10.1186/s40537-020-00298-6

DURILLO, Juan J.; NEBRO, Antonio J. jMetal: A Java framework for multi-objective optimization. **Advances in Engineering Software**, [*s. l.*], v. 42, n. 10, p. 760–771, 2011. Disponível em: https://doi.org/10.1016/j.advengsoft.2011.05.014. Acesso em: 7 fev. 2019.

DUTRA, Hiago Porto. **Implantação de Sistemas de Visualização Científica sob demanda utilizando a plataforma Amazon Web Services**. 42 f. 2018. - Universidade Federal de Uberlândia - UFU, [*s. l.*], 2018.

EAGER, Derek L.; ZAHORJAN, John; LAZOWSKA, Edward D. Speedup Versus Efficiency in Parallel Systems. **IEEE Transactions on Computers**, [*s. l.*], v. 38, n. 3, p. 408–

423, 1989. Disponível em: https://doi.org/10.1109/12.21127

ELKANO, Mikel *et al.* A global distributed approach to the Chi et al. fuzzy rule-based classification system for big data classification problems. *In*: , 2017, Naples-Italy. **2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. Naples-Italy: IEEE, 2017. p. 1–6. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2017.8015544

ELKANO, Mikel *et al.* CFM-BD: A Distributed Rule Induction Algorithm for Building Compact Fuzzy Models in Big Data Classification Problems. **IEEE Transactions on Fuzzy Systems**, [*s. l.*], v. 28, n. 1, p. 163–177, 2020. Disponível em: https://doi.org/10.1109/TFUZZ.2019.2900856

ELKANO, Mikel *et al.* CHI-BD: A fuzzy rule-based classification system for Big Data classification problems. **Fuzzy Sets and Systems**, [*s. l.*], v. 348, p. 75–101, 2018. Disponível em: https://doi.org/10.1016/j.fss.2017.07.003

ESHELMAN, Larry J. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. *In*: ELSEVIER (org.). **Foundations of Genetic Algorithms**. [*S. l.*]: Morgan Kaufmann, 1991. p. 265–283. Disponível em: https://doi.org/10.1016/b978-0-08-050684-5.50020-3

FAZZOLARI, Michela *et al.* A Review of the Application of Multiobjective Evolutionary Fuzzy Systems : Current Status and Further Directions. **IEEE Transactions on Control Systems Technology**, [*s. l.*], v. 21, n. 1, p. 45–65, 2013a. Disponível em: https://doi.org/10.1109/TFUZZ.2012.2201338. Acesso em: 7 fev. 2019.

FAZZOLARI, Michela *et al.* A study on the application of instance selection techniques in genetic fuzzy rule-based classification systems: Accuracy-complexity trade-off. **Knowledge-Based Systems**, [*s. l.*], v. 54, p. 32–41, 2013b. Disponível em: https://doi.org/10.1016/j.knosys.2013.07.011. Acesso em: 2 jul. 2014.

FERNANDEZ-BASSO, Carlos; RUIZ, M. Dolores; MARTIN-BAUTISTA, Maria J. Spark solutions for discovering fuzzy association rules in Big Data. **International Journal of Approximate Reasoning**, [*s. l.*], v. 137, p. 94–112, 2021. Disponível em: https://doi.org/10.1016/J.IJAR.2021.07.004

FERNANDEZ, Alberto *et al.* Revisiting Evolutionary Fuzzy Systems: Taxonomy,

applications, new trends and challenges. **Knowledge-Based Systems**, [*s. l.*], v. 80, p. 109–121, 2015. Disponível em: https://doi.org/10.1016/j.knosys.2015.01.013

FERNANDEZ, Alberto; ALMANSA, Eva; HERRERA, Francisco. Chi-Spark-RS: An Spark-built evolutionary fuzzy rule selection algorithm in imbalanced classification for big data problems. *In*: , 2017. **2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [*S. l.*]: IEEE, 2017. p. 1–6. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2017.8015520. Acesso em: 6 out. 2019.

FERNANDEZ, Alberto; RIO, Sara del; HERRERA, Francisco. A First Approach in Evolutionary Fuzzy Systems Based on the Lateral Tuning of the Linguistic Labels for Big Data Classification. *In*: , 2016. **2016 IEEE International Conference on Fuzzy Systems**. [*S. l.*]: IEEE, 2016. p. 1437–1444. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2016.7737858. Acesso em: 27 nov. 2016.

FERRANTI, Andrea *et al.* A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. **Information Sciences**, [*s. l.*], v. 415–416, p. 319–340, 2017. Disponível em: https://doi.org/10.1016/j.ins.2017.06.039. Acesso em: 24 abr. 2019.

FERRANTI, Andrea; MARCELLONI, Francesco; SEGATORI, Armando. A Multi-Objective Evolutionary Fuzzy System for Big Data. *In*: , 2016. **2016 IEEE International Conference on Fuzzy Systems**. [*S. l.*]: IEEE, 2016. p. 1562–1569. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2016.7737876. Acesso em: 28 nov. 2016.

FONSECA, Carlos M.; FLEMING, Peter J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *In*: , 1993, San Mateo - CA - USA. **Proceedings of the Fifth International Conference**. San Mateo - CA - USA: Morgan Kaufmann Publishers Inc., 1993. p. 416–423. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.9077&rep=rep1&type=pdf. Acesso em: 29 fev. 2016.

GENG, Xiaojiao; LIANG, Yan; JIAO, Lianmeng. Evidential Association Classification for High-Dimensional Data. **2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics, ICCCBDA 2021**, [*s. l.*], p. 100–105, 2021. Disponível em: https://doi.org/10.1109/ICCCBDA51879.2021.9442509

GHEMAWAT, Sanjay; GOBIOFF, Howard; LEUNG, Shun Tak. The google file system. *In*: , 2003. **Operating Systems Review (ACM)**. [*S. l.: s. n.*], 2003. Disponível em: https://doi.org/10.1145/1165389.945450

GOLDBERG, David E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. New York - NY - USA: Addison-Wesley, 1989. ISSN 0926-6003.v. 48*E-book*.

GOLDBERG, David E; HOLLAND, John Henry. Genetic Algorithms and Machine Learning. **Machine Learning**, [*s. l.*], v. 3, n. 2–3, p. 95–99, 1988. Disponível em: https://doi.org/10.1007/BF00113892. Acesso em: 1 set. 2014.

GOU, JinSend *et al.* **A new parallel FRBCS model based on Wang-Mendel and particle swarm optimization algorithms**. [*S. l.*], 2020. Disponível em: https://www-scopus.ez31.periodicos.capes.gov.br/record/display.uri?eid=2-s2.0-85091862672&origin=resultslist&sort=plf-f&src=s&st1=%2528%2522FRBCS%2522%2529+AND+%2522big+data%2522&sid=033ca323e706685f837dba769e1a447d&sot=b&sdt=b&sl=39&s=TITLE-ABS-KEY%2528%2528%2522FRBCS%2522%25. Acesso em: 25 fev. 2022.

HERRERA, F; LOZANO, M; VERDEGAY, JL. Applying genetic algorithms in fuzzy optimization problems. **Fuzzy Systems & Artificial Inteligence Reports and Letters**, [*s. l.*], v. 1, p. 39–52, 1994. Disponível em: http://www.researchgate.net/publication/2664250_Applying_Genetic_Algorithms_in_Fuzzy_Optimization_Problems/file/3deec525ad36c5e985.pdf. Acesso em: 20 set. 2014.

ITALIANO, Giuseppe F *et al.* Dynamic Algorithms for the Massively Parallel Computation Model. **CoRR**, [*s. l.*], v. abs/1905.09175, 2019. Disponível em: http://arxiv.org/abs/1905.09175

JAIMES, Antonio López; COELLO, Carlos A. Coello. Many-Objective Problems: Challenges and Methods. *In*: SPRINGER HANDBOOK OF COMPUTATIONAL INTELLIGENCE. Berlin, Heidelberg: Springer, 2015. p. 1033–1046. Disponível em: https://doi.org/10.1007/978-3-662-43505-2_51. Acesso em: 7 nov. 2016.

JAMSHIDI, Omid; PILEVAR, Abdol Hamid. Automatic Segmentation of Medical Images Using Fuzzy c-Means and the Genetic Algorithm. **Journal of Computational Medicine**, [*s. l.*], v. 2013, p. 1–7, 2013. Disponível em: https://doi.org/10.1155/2013/972970

JONG, Kenneth Alan De. **An Analysis of the Behavior of a Class of Genetic Adaptive Systems**. 268 f. 1975. - University of Michigan, [*s. l.*], 1975.

JONG, Kenneth Alan De. **Evolutionary Computation: A Unified Approach**. Cambridge - MA - USA: The MIT Press, 2006. *E-book*.

KHAN, M. Ali Ud Din; UDDIN, Muhammad Fahim; GUPTA, Navarun. Seven V's of Big Data understanding Big Data to extract value. *In*: , 2014. **Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education - "Engineering Education: Industry Involvement and Interdisciplinary Trends", ASEE Zone 1 2014**. [*S. l.: s. n.*], 2014. Disponível em: https://doi.org/10.1109/ASEEZone1.2014.6820689

KHAN, Nawsher *et al.* The 10 Vs, Issues and Challenges of Big Data. *In*: , 2018, New York, New York, USA. **Proceedings of the 2018 International Conference on Big Data and Education - ICBDE '18**. New York, New York, USA: ACM Press, 2018. p. 52–56. Disponível em: https://doi.org/10.1145/3206157.3206166. Acesso em: 20 set. 2019.

KNOWLES, J.; CORNE, D. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. *In*: , 1999. **Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)**. [*S. l.*]: IEEE, 1999. p. 98–105. Disponível em: https://doi.org/10.1109/CEC.1999.781913. Acesso em: 27 mar. 2017.

KOEHN, Philipp. **Combining Genetic Algorithms and Neural Networks : The Encoding Problem**. 1–67 f. 1994. [*s. l.*], 1994.

KUMAR, Manoj *et al.* Genetic Algorithm: Review and Application. **International Journal of Information Technology and Knowledge Management**, [*s. l.*], v. 2, n. 2, p. 451–454, 2010. Disponível em: http://www.csjournals.com/IJITKM/PDF 3-1/55.pdf. Acesso em: 19 set. 2014.

LANDSET, Sara *et al.* A survey of open source tools for machine learning with big data in the Hadoop ecosystem. **Journal of Big Data**, [*s. l.*], 2015. Disponível em: https://doi.org/10.1186/s40537-015-0032-1

LANEY, Doug. 3D Data Management: Controlling Data Volume, Velocity, and Variety. **Application Delivery Strategies**, [*s. l.*], 2001. Disponível em: https://doi.org/10.1016/j.infsof.2008.09.005

LICHMAN, M. **UCI Machine Learning Repository [http://archive.ics.uci.edu/ml].** [*S. l.*: *s. n.*], 2013.

LIMA, Helano Póvoas de; CAMARGO, Heloisa de Arruda. A Methodology for Building Fuzzy Rule-Based Systems Integrating Expert and Data Knowledge. *In*: , 2014, Sao Carlos - Sao Paulo. **2014 Brazilian Conference on Intelligent Systems**. Sao Carlos - Sao Paulo: IEEE, 2014. p. 300–305. Disponível em: https://doi.org/10.1109/BRACIS.2014.61. Acesso em: 22 mar. 2017.

LIN, Liduo; PAN, Li; LIU, Shijun. Methods for improving the availability of spot instances: A survey. **Computers in Industry**, [*s. l.*], v. 141, p. 103718, 2022. Disponível em: https://doi.org/10.1016/J.COMPIND.2022.103718

LINDEN, Ricardo. **Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional**. 3. ed. Rio de Janeiro - Brazil: Brasport, 2012. v. 3*E-book*.

LOPEZ, Victoria *et al.* On the use of MapReduce to build linguistic fuzzy rule based classification systems for big data. *In*: , 2014. **IEEE International Conference on Fuzzy Systems**. [*S. l.*]: IEEE, 2014. p. 1905–1912. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2014.6891753. Acesso em: 1 ago. 2019.

LUNA, Francisco; ALBA, Enrique. Parallel Multiobjective Evolutionary Algorithms. *In*: SPRINGER HANDBOOK OF COMPUTATIONAL INTELLIGENCE. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. p. 1017–1031. Disponível em: https://doi.org/10.1007/978-3-662-43505-2_50. Acesso em: 23 nov. 2016.

MACHADO, Jussara Gomes *et al.* A Modified NSGA-DO for Solving Multiobjective Optimization Problems. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, [*s. l.*], v. 13073 LNAI, p. 126–139, 2021. Disponível em: https://doi.org/10.1007/978-3-030-91702-9_9/COVER/. Acesso em: 4 jul. 2022.

MAMDANI, E H. Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis. **IEEE Transactions on Computers**, [*s. l.*], v. C–26, n. 12, p. 1182–1191, 1977. Disponível em: https://doi.org/10.1109/TC.1977.1674779. Acesso em: 29 ago. 2014.

MANN, J W; SMITH, G D. A Comparison of Heuristics for Telecommunications

Traffic Routing. *In*: RAYWARD-SMITH, V J *et al.* (org.). **Modern Heuristic Search Methods**. [*S. l.*]: Wiley, 1996. p. 235–254.

MENDEL, Gregor. Versuche über Pflanzenhybriden. **Verhandlungen des naturforschenden Vereins Bruenn**, [*s. l.*], 1866. Disponível em: https://doi.org/10.5962/bhl.title.61004

MENG, Xiangrui *et al.* MLlib: Machine Learning in Apache Spark. **Journal of Machine Learning Research**, [*s. l.*], v. 17, n. 34, p. 1–7, 2016. Disponível em: http://jmlr.org/papers/v17/15-237.html

MITCHELL, M. **An introduction to genetic algorithms**. Fifthed. London - England: MIT Press, 1998. *E-book*.

NGUYEN, H T; WALKER, E A. **A First Course in Fuzzy Logic**. Thirded. Boca Raton - FL - USA: Chapman & Hall/CRC, 2006.

PACHECO, Marco Aurélio Cavalcanti. **Algoritmos genéticos: princípios e aplicaçõesICA: Laboratório de Inteligência Computacional Aplicada -**. [*S. l.: s. n.*], 1999.

PARETO, Vilfredo. **Cours D'Economie Politique**. Paris - French: l'Université de Lausanne, 1896.

PARK, Jaehun *et al.* Operator allocation in cellular manufacturing systems by integrated genetic algorithm and fuzzy data envelopment analysis. **The International Journal of Advanced Manufacturing Technology**, [*s. l.*], 2014. Disponível em: https://doi.org/10.1007/s00170-014-6103-1. Acesso em: 23 set. 2014.

PIMENTA, Adinovam Henriques de Macedo. **Geração genética multiobjetivo de bases de conhecimento fuzzy com enfoque na distribuição das soluções não dominadas**. 111 f. 2014. - Federal University of São Carlos - UFSCar, [*s. l.*], 2014. Disponível em: https://repositorio.ufscar.br/handle/ufscar/8574

PIMENTA, Adinovam H. M.; CAMARGO, Heloisa de Arruda. NSGA-DO: Non-Dominated Sorting Genetic Algorithm Distance Oriented. *In*: , 2015, Istambul - Turkey. **2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. Istambul - Turkey: IEEE, 2015. p. 1–8. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2015.7338080. Acesso em: 22 mar. 2017.

REZENDE, Solange Oliveira. **Sistemas Inteligentes: Fundamentos e Aplicações**. 1. ed. Barueri - SP - Brazil: Manole, 2005.

RODRÍGUEZ-FDEZ, I.; MUCIENTES, M.; BUGARÍN, A. FRULER: Fuzzy Rule Learning through Evolution for Regression. **Information Sciences**, [*s. l.*], v. 354, p. 1–18, 2016a. Disponível em: https://doi.org/10.1016/j.ins.2016.03.012. Acesso em: 27 mar. 2017.

RODRÍGUEZ-FDEZ, I.; MUCIENTES, M.; BUGARÍN, A. S-FRULER: Scalable fuzzy rule learning through evolution for regression. **Knowledge-Based Systems**, [*s. l.*], 2016b. Disponível em: https://doi.org/10.1016/j.knosys.2016.07.034

RODRIGUEZ-MIER, Pablo; MUCIENTES, Manuel; BUGARÍN, Alberto. Feature Selection and Evolutionary Rule Learning for Big Data in Smart Building Energy Management. **Cognitive Computation**, [*s. l.*], p. 1–16, 2019. Disponível em: https://doi.org/10.1007/s12559-019-09630-6. Acesso em: 30 maio 2019.

ROSENBERG, Richard S. **Simulation of Genetic Populations with Biochemical Properties**. 232 f. 1967. - University of Michigan, [*s. l.*], 1967.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 3. ed. New Jersey - USA: Prentice-Hall, 2009. *E-book*.

SANTANA, Maykon Rocha. **Evolsys: Um Ambiente de Configuração e Análise de Algoritmos Evolutivos para Sintonia da Base de Regras Fuzzy do Sistema de Controle de um FMS**. 214 f. 2015. - Federal University of São Carlos - UFSCar, [*s. l.*], 2015. Disponível em: https://repositorio.ufscar.br/handle/ufscar/8413

SANTANA, Maykon Rocha; CAMARGO, Heloisa De Arruda. A parallel strategy for generation of fuzzy rule bases in big data problems using the NSGA-DO. **Proceedings - 2019 Brazilian Conference on Intelligent Systems, BRACIS 2019**, [*s. l.*], p. 48–53, 2019. Disponível em: https://doi.org/10.1109/BRACIS.2019.00018

SCHAFFER, J. David. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *In*: , 1985, Hillsdale - NJ. **Genetic Algorithms and their Applications: Proceedings of the 1st International Conference on Genetic Algorithms**. Hillsdale - NJ: Lawrence Erlbaum, 1985. p. 93–100. Disponível em: https://www.semanticscholar.org/paper/Multiple-Objective-Optimization-with-Vector-

Genetic-Schaffer/e51bc6ce7b1e19ff3f5b5386d2ca2b7e65eecfc3. Acesso em: 8 fev. 2019.

SCHAFFER, James David. **Some experiments in machine learning using vector evaluated genetic algorithms**. 166 f. 1984. - Vanderbilt University, [*s. l.*], 1984.

SEGATORI, Armando; MARCELLONI, Francesco; PEDRYCZ, Witold. On Distributed Fuzzy Decision Trees for Big Data. **IEEE Transactions on Fuzzy Systems**, [*s. l.*], v. 26, n. 1, p. 174–192, 2018. Disponível em: https://doi.org/10.1109/TFUZZ.2016.2646746. Acesso em: 24 abr. 2019.

SHLEEG, A A; ELLABIB, I M. Comparison of Mamdani and Sugeno Fuzzy Interference Systems for the Breast Cancer Risk. **International Journal of Computer, Information, Systems and Control Engineering**, [*s. l.*], v. 7, n. 10, p. 695–699, 2013. Disponível em: http://www.waset.org/publications/17193. Acesso em: 29 ago. 2014.

SILBERSCHATZ, Abraham.; GALVIN, Peter B.; GAGNE, Greg. **Operating system concepts**. Hoboken - NJ: John Wiley & Son ,Inc, 2012. *E-book*.

SILVA, Bhagya Nathali; DIYAN, Muhammad; HAN, Kijun. Big Data Analytics. *In*: SPRINGERBRIEFS IN COMPUTER SCIENCE. [*S. l.: s. n.*], 2019. Disponível em: https://doi.org/10.1007/978-981-13-3459-7_2

SINGH, Harpreet *et al.* Real-Life Applications of Fuzzy Logic. **Advances in Fuzzy Systems**, [*s. l.*], p. 1–3, 2013. Disponível em: https://doi.org/10.1155/2013/581879

SRINIVAS, N.; DEB, Kalyanmoy. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. **Evolutionary Computation**, [*s. l.*], v. 2, n. 3, p. 221–248, 1994. Disponível em: https://doi.org/10.1162/evco.1994.2.3.221. Acesso em: 8 fev. 2019.

SUDHOLT, Dirk. Parallel Evolutionary Algorithms. *In*: SPRINGER HANDBOOK OF COMPUTATIONAL INTELLIGENCE. Berlin, Heidelberg: Springer, 2015. p. 929–959. Disponível em: https://doi.org/10.1007/978-3-662-43505-2_46

SUN, Zhaohao. 10 Bigs : Big Data and Its Ten Big Characteristics. **Managerial Perspectives on Intelligent Big Data Analytics**, [*s. l.*], 2018. Disponível em: https://doi.org/10.13140/RG.2.2.31449.62566

TAKAGI, Tomohiro; SUGENO, Michio. Fuzzy Identification of Systems and Its

Applications to Modeling and Control. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, [*s. l.*], v. SMC-15, n. 1, p. 116–132, 1985. Disponível em: https://doi.org/10.1109/TSMC.1985.6313399. Acesso em: 29 ago. 2014.

TANENBAUM, Andrew S.; STEEN, Maarten Van. **Sistemas Distribuídos: Princípios e Paradigmas**. 2ª ed.ed. São Paulo - SP - Brazil: Pearson Education, 2007.

TSAKIRIDIS, Nikolaos L.; THEOCHARIS, John B.; ZALIDIS, George C. An evolutionary fuzzy rule-based system applied to real-world Big Data - the GEO-CRADLE and LUCAS soil spectral libraries. *In*: , 2018. **2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [*S. l.*]: IEEE, 2018. p. 1–8. Disponível em: https://doi.org/10.1109/FUZZ-IEEE.2018.8491489. Acesso em: 19 ago. 2019.

VOHRA, Deepak; VOHRA, Deepak. Apache HBase. *In*: PRACTICAL HADOOP ECOSYSTEM. [*S. l.: s. n.*], 2016a. Disponível em: https://doi.org/10.1007/978-1-4842-2199-0_4

VOHRA, Deepak; VOHRA, Deepak. Apache Hive. *In*: PRACTICAL HADOOP ECOSYSTEM. [*S. l.: s. n.*], 2016b. Disponível em: https://doi.org/10.1007/978-1-4842-2199-0_3

VOHRA, Deepak; VOHRA, Deepak. Apache Sqoop. *In*: PRACTICAL HADOOP ECOSYSTEM. [*S. l.: s. n.*], 2016c. Disponível em: https://doi.org/10.1007/978-1-4842-2199-0_5

ZAHARIA, Matei *et al.* Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *In*: , 2012, San Jose - CA - USA. **NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation**. San Jose - CA - USA: UNSENIX, 2012. p. 15–28. Disponível em: https://doi.org/10.1111/j.1095-8649.2005.00662.x

ZAHARIA, Matei *et al.* Spark: Cluster computing with working sets. [*s. l.*], 2010. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.180.9662. Acesso em: 11 jul. 2022.

ZECEVIC, Petar; BONACI, Marko. **Spark in Action**. Shelter Island - New York - EUA: Manning Publications, 2017. *E-book*.

ZHOU, Chi. Fast Parallelization of Differential Evolution Algorithm Using MapReduce. *In*: , 2010, New York, New York, USA. **Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10**. New York, New York, USA: ACM Press, 2010. p. 1113. Disponível em: https://doi.org/10.1145/1830483.1830689. Acesso em: 20 mar. 2017.

ZHOU, Lina *et al.* Machine Learning on Big Data: Opportunities and Challenges. **Neurocomputing**, [*s. l.*], v. 237, p. 350–361, 2017. Disponível em: https://doi.org/10.1016/j.neucom.2017.01.026. Acesso em: 20 mar. 2017.

ZITZLER, Eckart; LAUMANNS, Marco; THIELE, Lothar. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. **Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems**, [*s. l.*], 2001. Disponível em: https://doi.org/10.1.1.28.7571