Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciência
Faculdade de Engenharia

Adriano Valladão de Barros

Síntese de Árvores de Padrões Fuzzy Através de Busca Participativa

Rio de Janeiro
2018

Adriano Valladão de Barros

Síntese de Árvores de Padrões Fuzzy Através de Busca Participativa

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientador: Prof. Dr. Jorge Luís Machado do Amaral

Rio de Janeiro

2018

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

Assinatura

Data

Adriano Valladão de Barros

Síntese de Árvores de Padrões Fuzzy Através de Busca Participativa

Dissertação apresentada como requisito parcial para obtenção do título de Mestre, ao programa de Pós-Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em:

Banca Examinadora:

_____

Prof. Dr. Jorge Luís Machado do Amaral (Orientador)
Faculdade de Engenharia - UERJ

_____

Prof. Dr. Douglas Mota Dias
Faculdade de Engenharia - UERJ

_____

Profa. Dra. Karla Tereza Figueiredo Leite
Instituto de Matemática e Estatística - UERJ

_____

Prof. Dr. Ronaldo Ribeiro
Instituto Militar de Engenharia - IME

Rio de Janeiro
2018

DEDICATÓRIA

Dedico este trabalho aos meus pais.

AGRADECIMENTOS

A gratidão em diversos momentos não é expressa da maneira como o coração intui. Faltam palavras. Elas se tornam limitadas diante do que era para ser expresso, esbarrando nos limites da linguagem ou simplesmente a inconsciência de certos momentos impendem que ela seja expressa de maneira mais sincera possível, sem amarras, bloqueios ou traumas. Expressar gratidão recria a conexão que a mente tira, quando a expressamos, nos aproximamos mais do nosso estado natural consciente. A gratidão nos reconecta. Espero que consiga da melhor maneira expressar pessoalmente a gratidão que escrevo nesta página, não por essa dissertação, mas pelas vivências e percepções em decorrência dela.

Agradeço aos meus pais que possibilitaram a estabilidade social, sob diversos aspectos, para que algumas escolhas fossem possíveis ao longo da vida. Agradeço aos sacrifícios que fizeram durante muitos anos. Agradeço com todo meu amor a todo o amor presente, independente da materialidade dos problemas e inconsciências da rotina.

Agradeço ao meu orientador Jorge Amaral que inspirou e viabilizou a realização de diversos feitos acadêmicos, estimulando a superação de algumas barreiras e tornando possível a realização deste trabalho. Grato pela parceria, ensinamentos e paciência.

Agradeço ao amigo André que se tornou muito presente durante essa fase, ajudando de diversas formas, seja numa simples conversa sincera e consciente ou lutando pelos amigos da forma que fosse preciso. Agradeço também às amizades que me possibilitou nesse período. Obrigado amigo.

Agradeço à Raquel, que me deu vivências sinceras de amor e liberdade durante essa fase. Mostrando o quanto uma relação livre de passado e futuro se torna amorosa e nos conecta com nosso próprio ser. A consciência que se tornou presente nesse período, se liga diretamente ao fechamento deste ciclo de forma harmoniosa e serena.

Agradeço aos amigos Joelmir, Diego e Leonardo pela amizade presente de outras épocas. Durantes muitos anos ela me permitiu tomar consciência de algumas perspectivas importantes. Agradeço de coração a cada um por todos os momentos, oportunidades vivenciadas e pela amizade fraterna. A união da nossa amizade vai além da presença física. Sou muito grato a vocês amigos.

Agradeço aos amigos e companheiros de mestrado Sávio, Noemi e Patrícia. Através de muito apoio e incentivo mútuo, juntos conseguimos seguir até o final.

Aos companheiros de laboratório Hugo, Everton, Anderson, Carlos, Mirian, George e Leandro pela receptividade, pelas trocas de conhecimento e reciprocidade no convívio.

Agradeço aos professores, funcionários e seguranças do laboratório LARISA, do Programa de Pós-Graduação em Engenharia Eletrônica e do Departamento de Engenharia Eletrônica e de Telecomunicações.

Agradeço a todos de coração.

Agradeço também à Capes e Faperj pelo apoio prestado neste projeto.

“... porque toda a sua atividade serve apenas para livrá-lo da sua loucura, é uma catarse.”

Osho

RESUMO

BARROS, Adriano V. Síntese de Árvores de Padrões Fuzzy através de Busca Participativa. 2018. 90f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

Este trabalho tem como objetivo apresentar um sistema que sintetiza classificadores interpretáveis baseados nos modelos *fuzzy*. A estrutura *fuzzy* utilizada é intitulada Árvores de Padrão Fuzzy (APF) que não faz uso de regras e surge como uma opção ao Sistema Fuzzy Baseado em Regras. São propostas algumas alternativas aos métodos já desenvolvidos na síntese dessas estruturas, utilizando a Aprendizagem Participativa no processo de busca das melhores árvores para a tarefa de classificação em um processo supervisionado de treinamento. Na estrutura em árvore, os nós internos são operadores lógicos generalizados já utilizados em sistemas *fuzzy* e as folhas são termos fuzzy associados a um atributo de entrada. O método gera uma estrutura em árvore para cada classe, permitindo uma avaliação hierárquica da influência para cada atributo na tarefa de classificação. A partir desta estrutura, é possível obter uma expressão que possibilita uma boa interpretabilidade unida a valores de acurácia que estão no nível do estado-da-arte. Os métodos de síntese das árvores propostos aqui são baseados no processo de busca participativa, sendo este fundamentado em uma população que busca sua evolução através de gerações, orientada por um método de avaliação de cada indivíduo que compõe essa população. No método participativo a busca prossegue orientada pela compatibilidade entre indivíduos, sempre mantendo o melhor indivíduo nas populações posteriores e introduzindo indivíduos aleatoriamente em cada passo do algoritmo. Os métodos de síntese propostos têm finalidade de explorar melhor o espaço de busca baseado no bom desempenho que a busca participativa apresenta em relação ao estado-da-arte. Os métodos foram comparados com os algoritmos de Máquinas de Vetores de Suporte, K Vizinhos mais próximos, Florestas Aleatórias e com o método de aprendizado originalmente proposto das APF, em diversas bases de dados do UCI Machine Learning Repository. Observou-se que os modelos são compatíveis com o estado-da-arte, gerando resultados competitivos e com árvores menores, que melhoram os níveis de interpretabilidade.

Palavras-Chave: Aprendizado de máquinas, Sistemas Fuzzy, Árvores de Padrões Fuzzy, Modelo Hierárquico, Busca Participativa, Aprendizado Participativo, Classificação, Interpretabilidade.

ABSTRACT

BARROS, Adriano V. Synthesis of Fuzzy Pattern Trees through Participatory Search. 2018. 90f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

The aim of this work is to introduce a system to synthesize the interpretable classifiers based on fuzzy modeling. The structure fuzzy used is named Fuzzy Pattern Trees (FPT), which does not use rules and comes up as an option to the Rule-Based Fuzzy Systems. Some alternatives are proposed to methods already known of synthesis for these structures, they are using Participatory Learning on the search process of better trees to the classification task in a supervised process of training. In the tree structure, the inner nodes are generalized logic operators already in use on Fuzzy Systems and the leaf nodes are fuzzy terms associated to an input attribute. The method generates a tree structure for each class, allowing a hierarchical evaluation of the influence for each attribute in the classification task. From this structure, it is possible to obtain an expression that allows a good interpretability linked to the values of accuracy that are at the state-of-the-art level. The methods of tree synthesis here proposed are based on participatory search, it is grounded on a population that seeks its evolution through generations, it is guided by a method of evaluation for each individual who belongs to this population. In the participatory method, the search continues guided by the compatibility between individuals, always keeping the best individual from the populations back and randomly introducing individuals at each step of the algorithm. The proposed methods of synthesis have the purpose of exploring the best space of search based on the good performance that the participatory search presents in relation to the state-of-the-art. The methods were compared with the algorithms: Support Vector Machine, k-nearest neighbors, Random Forest and the learning method originally proposed by the FPT, in several UCI Machine Learning databases. It was observed that the models are compatible with the state-of-the-art and smaller trees, generating competitive results, that improve the levels of interpretability.

Keywords: Machine Learning, Fuzzy Systems, Fuzzy Pattern Trees, Hierarchical Model, Participatory Search, Participatory Learning, Classification, Interpretability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

AG	Algoritmo genético
AP	Aprendizagem Participativa
APF	Árvore de Padrão Fuzzy
AUC	*Area Under the ROC curve*
BBP	Busca Baseada em População
BP	Busca Participativa
CA	Curvas de Aprendizado
CE	Computação Evolucionária
DPSA	*Diferential Participatory Search with Arithmetical Recombination*
DPST	*Diferential Participatory Search with Selective Transfer*
ED	Evolução Diferencial
K-NN	*K Nearest Neighbor*
LDM	*Lista Diferencial de Movimentos*
OWA	*Ordered Weighted Average*
OC	Otimização Combinatória
PGC	Programação Genética Cartesiana
PSAR	*Participatory Search with Arithmetical Recombination*
PSST	*Participatory Search with Selective Transfer*
RMSE	*Root Mean Squared Error*
ROC	*Receiver Operating Characteristic*
RF	*Random Forest*
RSVM	*Radial Support Vector Machine*
SFBR	Sistemas Fuzzy Baseado em Regras
TS	Transferência Seletiva
TSP	*Traveling Salesman Problemam*
VC	Validação Cruzada

SUMÁRIO

Introdução

Na rotina de grande parte da população é necessária uma considerável habilidade para lidar com um crescente número de informações, o que afeta o processo de tomada de decisão tanto a nível pessoal quanto em ambientes profissionais, financeiros ou governamentais. O processo de tomada de decisão é um atributo exclusivamente humano que é influenciado diretamente por diversos aspectos da construção social da pessoa, junto a fatores que envolvem sentimentos e instintos que, além de complexos, estão vinculados a níveis inacessíveis do subconsciente. O que claramente torna o ser humano um ser complexo em diversos níveis do ponto de vista pragmático. Toda complexidade na análise da mente humana se torna extremamente interessante diante do fato dela ser um mecanismo biológico que tem a capacidade de compreender e deduzir como nenhum outro mecanismo conhecido. Deste modo, surgem diversos ramos de pesquisa que tentam entender os processos que ocorrem na mente sob diversas perspectivas, tentando não só compreender mas transcender sua capacidade em lidar com informações e otimizar decisões. Um deles tem fundamentado sua pesquisa em torno do entendimento do processo de tomada de decisão, dentro da grande área de inteligência artificial com o auxílio de conceitos, técnicas e ferramentas de aprendizado de máquinas e mineração de dados. O aprendizado de máquinas trata do estudo de modelos que podem aprender a partir de um conjunto de dados (WITTEN; FRANK, 2005). A mineração de dados trata da extração automática de padrões que representam o conhecimento armazenado em grandes bases de dados (HAN; KAMBER, 2006). Em outras palavras, representa o processo de extração de informações implícitas, previamente desconhecidas e potencialmente úteis a partir de dados. A ideia seria a de se construir programas que consigam explorar bases de dados automaticamente, em busca de regularidades ou padrões. Padrões fortes encontrados provavelmente podem ser generalizados para que se façam previsões sobre dados futuros (BARBOSA, 2012).

Nesses processos de decisão, as informações e conhecimento prévio, que envolvem a questão em análise, ficam muitas vezes obscurecidas em relação aos seus níveis de significância, apesar de decisões acertadas serem tomadas. Um exemplo disso são casos em que pessoas conseguem tomar decisões pertinentes sem que consigam apontar com certa exatidão quais fatores influenciaram naquela decisão. Assim como no exemplo, existem modelos matemáticos com diversificas heurísticas que, da mesma forma, criam métodos de previsão onde não deixam claro como a decisão foi tomada. A

extração de informação no processo de tomada de decisão se põe como importante sob diversas perspectivas, pois descreve como um método de previsão chegou a determinada conclusão. Essa interpretabilidade do método fornece suporte à extração de conhecimento, como, por exemplo, na área de diagnósticos e prognósticos ou detecção e identificação de fraudes dentre outras. Quando se deseja aliar a previsão com o entendimento da tomada de decisão, surgem algumas alternativas, como os modelos gerados por abordagens simbólicas. Os modelos de tal abordagem procuram unir sua interpretabilidade com níveis de acurácia competitivos. Alguns métodos mais bem-sucedidos aplicados na síntese de modelos interpretáveis são os baseados na teoria dos conjuntos *fuzzy*, pois esses modelos criam uma interface expressa em linguagem natural.

São encontradas diversas ferramentas consolidadas que podem ser utilizadas na tomada de decisão, tais como: Redes Neurais Artificiais (SCHMITZ; ALDRICH; GOUWS, 2000), as Árvores de Decisão (BREIMAN et al., 1984), Lógica Fuzzy (ZADEH, 1996), os Algoritmos Genéticos (HOLLAND, 1975), e muitas outras. Cada uma dessas técnicas possui particularidades e podem ser combinadas umas às outras de forma híbrida, como no caso de Sistemas Fuzzy-Genéticos (KARR, 1991), (THRIFT, 1991), (CORDON et al., 2001) ou Sistemas Neuro-Fuzzy (SUN; JANG, 1993), (NAUCK; KLAWONN; KRUSE, 1997), (ABRAHAM, 2005). Na lógica *fuzzy,* que surgiu a partir da Teoria dos Conjuntos Fuzzy (ZADEH, 1965), temos um sistema que pode ser visto como uma extensão da lógica convencional, mas possui a capacidade de tratar de maneira efetiva os problemas de representação de conhecimento em um ambiente de incertezas e imprecisões (ISHIBASHI, 2013). Ela é uma forma de lógica que tem como fundamento a aproximação e não a exatidão do conhecimento, admitindo infinitos valores intermediários entre 0 e 1 (PEDRYCZ; GOMIDE, 1998).

A Teoria dos Conjuntos Fuzzy é um dos paradigmas mais importantes da Inteligência Computacional, onde são explorados aspectos de inferência e de representação do conhecimento. A Teoria dos Conjuntos Fuzzy cria uma interface entre padrões quantitativos e estruturas de conhecimento qualitativas expressas em termos de linguagem natural. Essa característica faz com que ela seja atraente do ponto de vista da representação do conhecimento, permitindo que o conhecimento adquirido em uma base de dados possa ser representado de uma forma linguística compreensível, gerando uma maior interpretabilidade do modelo (HÜLLERMEIER, 2005). Podemos encarar a interpretabilidade como sendo a capacidade de expressar o comportamento de um sistema de forma compreensível. Ela é uma propriedade subjetiva e normalmente está ligada a

vários fatores que estão relacionados a estrutura do modelo, tais como: número de variáveis de entrada, o número de regras, o número de termos linguístico, etc. Além de não existir uma medida padronizada para avaliar a interpretabilidade (GACTO; ALCALÁ; HERRERA, 2011), ela é antagônica à acurácia, exigindo que o pesquisador venha a buscar um equilíbrio entre essas duas métricas (CASILLAS, 2003).

A partir da Teoria dos Conjuntos Fuzzy, podem ser sintetizados os Sistemas Fuzzy Baseados em Regras (SFBR) que são sistemas que consideram regras na forma *if-then* cujos antecedentes e consequentes são compostos por variáveis *fuzzy* (CORDON et al., 2001). Neles a representação do conhecimento é realizada através de variáveis que utilizam termos linguísticos. Devido a essa característica, os SFBR têm sido aplicados em uma grande variedade de problemas no qual as incertezas e imprecisões são encontradas em diferentes formas (ISHIBASHI, 2013). Os SFBR podem representar tanto as funções de classificação quanto as de regressão e existe um grande número de estratégias que foram desenvolvidas para induzir modelos *fuzzy* baseados em regras (CORDÓN, 2011). A obtenção de um sistema *fuzzy* baseado em regras pode não ser uma tarefa simples. Se o número de variáveis for grande, o número possível de regras aumenta exponencialmente, tornando o processo de busca pelo conjunto adequado de regras, mais difícil e gerando problemas de escalabilidade. Este efeito é conhecido como a "maldição da dimensionalidade". Além disso, dependendo da aplicação, uma grande quantidade de regras pode ser necessária para que o sistema atinja o desempenho desejado, por exemplo, em termos de acurácia (SANTOS, 2014).

Muitos algoritmos de aprendizagem de SFBR sofrem da maldição da dimensionalidade, fazendo com que o tempo computacional do algoritmo aumente de forma considerável e indesejável com o aumento no número de atributos, dificultando a construção do modelo (LEGARDA, 2016). Diversas alternativas foram propostas para lidar com essa questão, tais como: identificação de relações entre as variáveis para reduzir o número de variáveis usadas, combinação de duas ou mais variáveis para obter uma nova variável para substituir as variáveis originais, interpolação das regras e sistemas *fuzzy* hierárquicos (TORRA, 2002).

Em sistemas *fuzzy*, se um sistema possui n entradas e para cada entrada são definidos m conjuntos *fuzzy*, então o número total de regras necessário para a implementação de um sistema *fuzzy* completo será m^n, ou seja, se torna um problema exponencial. Para tornar esse problema tratável, sistemas *fuzzy* hierárquicos foram propostos, possibilitando um aumento linear do número de regras com incremento do

número de entradas (ROISENBERG, 1998). Na literatura existem diversos sistemas hierárquicos, tais como em (GONCALVES et al., 2006), (ZHANG, et al 2014), (BAYDOKHT, et al 2016), (CHANG, et al 2018). Eles caracterizam-se por possuir em diversos módulos que contribuem para a solução final. Os módulos de mais baixa ordem recebem como entradas algumas das variáveis e suas saídas são usadas pelos módulos de mais alta ordem para calcular a solução final. Esta hierarquia pode facilitar a interpretabilidade, pois os módulos de mais baixa ordem encapsulam apenas uma parte do conhecimento do sistema, sendo mais fáceis de compreender do que um único sistema monolítico de regras. Além disso, um sistema hierárquico pode auxiliar na escolha do compromisso entre interpretabilidade e acurácia, pois permitem a definição de módulos de mais baixa ordem de acordo com os critérios dos projetistas (SANTOS, 2014).

Em (HUANG; GEDEON; NIKRAVESH, 2008) foi proposto um novo modelo hierárquico com estrutura em árvore para indução de classificadores *fuzzy*, intitulado como Árvore de Padrões Fuzzy (APF). Esse modelo se torna atraente do ponto de vista da interpretabilidade pois a partir da estrutura utilizada no modelo é possível avaliar qual atributo de entrada se torna mais relevante no resultado final, além de possibilitar uma análise lógica de tomada de decisão (SENGE; HULLERMEIER, 2011). Na estrutura em árvore do modelo, suas folhas são termos *fuzzy* associados aos atributos e os nós internos são os operadores utilizados em sistemas *fuzzy*. Basicamente, esse algoritmo programa uma função que mapeia uma combinação de atributos de entrada em um número no intervalo [0,1] (ALMEIDA, 2017). O algoritmo original para o aprendizado dessas árvores, chamado por Bottom-up Induction (HUANG; GEDEON; NIKRAVESH, 2008), apresenta problemas de convergência prematura, podendo ficar preso em ótimos locais. Posteriormente, em (SENGE; HULLERMEIER, 2011), surge a possibilidade de melhoria das soluções obtidas para o mecanismo de busca global utilizado o método Top-Down Induction que utiliza a estratégia de busca denominada Beam Search (ZHANG, 1998). Essa estratégia explora um grafo de soluções expandindo o nó mais promissor em um conjunto limitado de opções. Este algoritmo possui uma característica “gulosa”, a busca no espaço de soluções é feita de forma restrita. Em (SANTOS, 2014) e (ALMEIDA, 2017) foram desenvolvidos métodos de indução de APF nas tarefas de classificação e regressão através da Programação Genética Cartesiana (PGC) (MILLER, 2011). Esses trabalhos tiveram como objetivo explorar melhor o espaço de busca, resultando em modelos competitivos em termos de acurácia e interpretabilidade. Nos trabalhos gerou-se árvores mais compactas com níveis de acurácia comparáveis aos dos primeiros dois

métodos desenvolvidos em (HUANG; GEDEON; NIKRAVESH, 2008) e (SENGE; HULLERMEIER, 2011) com um menor número de avaliações.

Em (SANTOS, 2014) foi utilizada como representação das APF uma lista de números inteiros com restrições, se enquadrando com a PGC de maneira conveniente. Essa representação se encaixa com a classe de problemas combinatórios. Baseado nesse tipo de problemas, a busca por alternativas de métodos de otimização do espaço de busca se torna conveniente. Dentro desse contexto, a Aprendizagem Participativa se torna interessante dados os resultados promissores alcançados recentemente.

A Aprendizagem Participativa (AP) foi introduzida no início dos anos 90 no domínio dos sistemas adaptativos e de aprendizagem (YAGER, 1990). A AP é basicamente um esquema de aprendizagem em que o processo de aprendizagem depende do que já é conhecido ou acreditado (LIU; GOMIDE, 2013). Dentro da área de AP temos a Aprendizagem Participativa Evolutiva, sendo composta por métodos evolutivos baseados em população em que essa população influencia a aptidão dos indivíduos durante a evolução, utilizando os conceitos da Computação Evolutiva. Esse paradigma se beneficia de conceitos da AP, Algoritmos Genéticos (AG) (HOLLAND, 1975) e da Evolução Diferencial (ED) (PRICE et al., 1997). Dentro da área de pesquisa da AP, em 2013 foi introduzido o Algoritmo de Aprendizado Genético Participativo (*Participatory Genetic Learning Algorithm* - PGLA) por (LIU; GOMIDE, 2013) e em 2016 o Algoritmo de Busca Participativa (*Participatory Search Algorithm* - PSA) em (LIU; GOMIDE, 2016).

Este trabalho apresenta um método para indução de modelos de Árvores de Padrões Fuzzy (APF) de forma automática, utilizado na tarefa de classificação. O método de aprendizado do APF foi substituído pela Busca Participativa. Foram realizados diversos estudos de casos, utilizando bases de dados artificiais e reais disponíveis na UCI *Machine Learning Repository* para se obter uma melhor compreensão do funcionamento do método e desenvolver estratégias para uma melhor generalização, avaliando o desempenho dos classificadores gerados.

O restante dessa dissertação está organizado da seguinte forma: O Capítulo 1 apresenta conceitos que definem a Busca Participativa e o método aplicado neste trabalho; o Capítulo 2 introduz as Árvores de Padrões Fuzzy. O Capítulo 3 discorre sobre problemas combinatórios e algoritmos para tratá-los; o Capítulo 4 apresenta os modelos propostos, o Capítulo 5 discute sobre os resultados obtidos e o Capítulo 6 apresenta a conclusão e a sugestão de trabalhos futuros.

1 Busca Participativa

O processo de busca é útil em inúmeras aplicações, tais como otimização de funções, aprendizado de máquina, processamento da informação e recuperação da informação. A busca por uma solução em um espaço de soluções pode se dar de diversas maneiras. Uma possível estratégia é a Busca Baseada em População (BBP). Técnicas que seguem esta abordagem geralmente se iniciam com uma população de indivíduos espalhados pelo espaço de soluções. O objetivo é que esta população evolua em uma sequência de gerações, até encontrar a melhor solução. Exemplos de técnicas que empregam esta abordagem são os Algoritmos Genéticos (AG) (HOLLAND, 1975), a Programação Genética (PG) (KOZA, 1992), a Otimização por Enxame de Partículas (EBERHART; KENNEDY, 1995) ou a Otimização por Colônia de Formigas (DORIGO; DI CARO, 1999), que estão dentro do ramo de pesquisa da Computação Evolucionária (CE). Uma maneira de abordar o papel que a própria população desempenha na evolução é a Aprendizagem Participativa (AP) (YAGER, 1990). A hipótese básica da AP é que a aprendizagem ocorre no âmbito do que já foi aprendido ou do que se acredita que seja verdadeiro. A implicação dessa abordagem é que todos os aspectos do processo de aprendizagem são afetados e guiados pelo conhecimento que se possui até o presente momento.

Neste capítulo serão apresentados alguns conceitos básicos e definições que envolvem os algoritmos de Busca Participativa (BP), que se constrói dentro da área de pesquisa da AP.

1.1 Conceitos básicos

No caso dos algoritmos baseados no paradigma da AP, a busca é guiada pela compatibilidade entre indivíduos de uma população, sempre mantendo o melhor indivíduo das populações posteriores e introduzindo novos indivíduos de forma aleatória em cada etapa do algoritmo.

Um fator fundamental da AP é o grau de compatibilidade entre a informação de entrada e o conhecimento atual. Uma questão importante é que nenhuma facilidade é fornecida para medir a confiança que temos na estrutura de conhecimento atual. Isso significa que à medida que avançamos no aprendizado é como se tivéssemos confiança

completa em nosso conhecimento atual. Se uma longa sequência de entradas tem pouca compatibilidade com o conhecimento atual, podemos acreditar que o que foi aprendido até agora é errado. Isso é visto como uma forma de excitação que é definida pela compatibilidade e por um mecanismo que monitora essa compatibilidade durante a aprendizagem, o índice de excitação. A Figura 1 resume essa ideia, onde o conhecimento atual, denotado por v(t), fornece um padrão contra o qual a informação de entrada z(t) é comparada durante a adaptação e afeta diretamente o processo de aprendizagem. A ligação entre a adaptação e excitação indica que a comparação entre v(t) e z(t) define como o sistema aceita e processa à informação de entrada em comparação a informação já conhecida. Isso corresponde à natureza participativa do processo de aprendizagem (LIU, 2016).

Um exemplo para ilustrar esse processo seria utilizar a AP para encontrar o mínimo de uma função. Nesse exemplo o conhecimento atual é representado pelo indivíduo que apresenta o menor valor na função e a informação de entrada como sendo a nova população da geração seguinte. A partir das etapas evolutivas da busca, a nova população é comparada com o melhor indivíduo da população anterior (conhecimento que se tem do problema até o momento). Se muitos novos indivíduos, quando comparados ao melhor indivíduo anterior, apresentam uma diferença grande, acredita-se que o mínimo encontrado até o momento não representa uma tendência de convergência para o mínimo da função. Assim, o mecanismo de excitação identifica essa diferença e controla o processo evolutivo de acordo com os operadores escolhidos durante as etapas de busca. Essa diferença, que é estabelecida pela compatibilidade entre o conhecimento atual e a informação de entrada atual, indica um ambiente favorável na direção do aprendizado.

A AP introduz um mecanismo de excitação para monitorar o desempenho do processo de aprendizagem, observando os valores do grau de compatibilidade do conhecimento atual com os insumos.

Figura 1 – Aprendizado Participativo (Adaptado de LIU 2016)

É bem sabido que a compatibilidade genética é um mecanismo pelo qual os indivíduos obtêm benefício da aptidão por meio da escolha do parceiro (AGBALI et al., 2010). A segurança reprodutiva e os benefícios da autocompatibilidade também podem ser fortemente influenciados pela densidade, diversidade ou tamanho da população (BUSH, 2005). Essas observações sugerem a possibilidade de construir algoritmos de BP nos quais a população desempenha um papel na determinação da reprodução e seu caminho evolutivo subsequente. Nesse sentido, a aprendizagem pode ser vista como uma espécie de aprendizado da aptidão, uma vez que a aptidão molda a adequação reprodutiva dos indivíduos. Quando a população afeta a evolução, a adequação de cada indivíduo resulta da combinação de seu próprio objetivo e sua compatibilidade com diferentes indivíduos. Em ambientes naturais, muitas vezes, o efeito da população na capacitação é tentar tornar os indivíduos mais parecidos com a própria população. (YAGER, 2000).

Em (LIU, 2016), é apresentada uma coleção de procedimentos de aprendizagem construídos com base na CE e na AP. O trabalho introduziu uma nova classe de algoritmos de pesquisa baseados em população usando operadores participativos de seleção, transferência seletiva, recombinação e mutação. Esses operadores atuam em conjunto com os graus de compatibilidade e a função objetivo moldando a avaliação da dependência dos indivíduos que governam a evolução da população. A classe de algoritmos de BP abordada no trabalho apresenta quatro instâncias, sendo elas: Busca Participativa com Transferência Seletiva (*Participatory Search with Selective Transfer* - PSST); Busca Participativa com Recombinação Aritmética (*Participatory Search with Arithmetical Recombination* - PSAR); Busca Participativa Diferencial com Transferência Seletiva (*Differential Participatory Search with Selective Transfer* - DPST) e Busca Participativa Diferencial com Recombinação Aritmética (*Differential Participatory Search with Arithmetical Recombination* - DPSA). Eles são distinguidos pela natureza da recombinação e pela ordem na qual as operações de seleção, recombinação e mutação são processadas em cada geração.

O trabalho de (LIU, 2016) foi desenvolvido centrando a pesquisa no PSAR devido aos resultados computacionais anteriores em (LIU; GOMIDE, 2016) mostrarem seu melhor desempenho em relação às outras instâncias. Os resultados computacionais apresentados revelam que o algoritmo é competitivo, computacionalmente simples e eficiente. Do ponto de vista da qualidade das soluções que ele desenvolve, o algoritmo é mais rápido e requer menos hiperparâmetros, tornando-o muito atraente na prática. Também foram desenvolvidas aplicações do PSAR na modelagem de sistemas baseados

em regras *fuzzy* com o objetivo de ilustrar potenciais aplicações da BP usando dados reais e comparar com resultados relatados na literatura. Os resultados corroboram que o algoritmo funciona melhor do que a abordagem do Sistema Fuzzy Genético (CORDON´N et al., 2001) (HERRERA, 2008) do estado da arte.

Na seção seguinte será descrito com mais detalheis o algoritmo Busca Participativa com Recombinação Aritmética (PSAR).

1.2 Busca Participativa com Recombinação Aritmética - PSAR

A análise de convergência e o comportamento do PSAR usa a teoria da busca aleatória (SOLIS; WETS, 1981). A técnica de busca aleatória oferece uma maneira de modelar o comportamento dos algoritmos. A seguir é apresentado o algoritmo que serve como uma descrição conceitual do PSAR que possibilita uma visão geral do algoritmo, posteriormente é apresentado um detalhamento dos seus procedimentos. No procedimento a seguir, $best^t$ representa o melhor indivíduo conhecido, S^t representa a população da geração *t*, $S^{t\prime}$ é uma população gerada, baseada na compatibilidade entre os indivíduos de S^t, μ_t é uma distribuição de probabilidade uniforme, p_r^t são os indivíduos recombinados, p_m^t são os indivíduos mutados, α o índice de excitação, ρ a compatibilidade e $p_{selected}$ é o grupo de selecionados, gerados a partir da avaliação dos indivíduos de S^t e $S^{t\prime}$ que mais se aproximam do $best^t$.

1: **procedimento** PSAR
2: Definir $t = 0$
3: Escolher $best^t$
4: **repetir**
5: Gerar S^t usando μ_t
6: Avaliar $S^{t'}$
7: Encontrar $best^t$ em S^t
8: Definir $best(S^t) = D(S^t, best^t)$
9: Definir $t = t + 1$
10: **até** critério de parada ser alcançado
11: **fim do procedimento**

Sendo a função *D,* indicada no PSAR conceitual, representada por:

$$D(S^t, best^t) = \begin{cases} p_r^t = (1 - \alpha\rho_r{}^t)s^t + (\alpha\rho_r{}^t)s^{t'} \\ p_m^t = best^t + \rho_m{}^t(p_{selected}^t - p_r^t) \\ best^t \end{cases} \quad (1)$$

Assumimos S^t como um conjunto de *N* indivíduos como *strings* de comprimento *n* na etapa *t* (passos do algoritmo). Inicia-se uma população S^t em $t = 0$ com *N* indivíduos aleatoriamente escolhidos e, para cada indivíduo de S^t, o indivíduo mais compatível entre os remanescentes é escolhido para montar a população $S^{t'}$ com os indivíduos *N*. S^t e $S^{t'}$ formam os grupos de recombinação. O próximo passo calcula o melhor indivíduo na população atual S^t, que recebe a notação $best$. A seleção prossegue, selecionando, entre cada indivíduo de S^t e o correspondente em $S^{t'}$, aquele que está mais próximo do $best$ analisando a compatibilidade entre eles e o $best$. A recombinação é feita de forma pareada entre os indivíduos do grupo de recombinação, ponderada pelos valores de compatibilidade e excitação. A mutação usa os indivíduos selecionados e recombinados para produzir variações cuja quantidade é ponderada pela compatibilidade e excitação. Se um recombinado for melhor do que o $best$ atual, ele o substitui, já se um indivíduo resultante da mutação for melhor do que o $best$ atual, ele passa ser o novo $best$. Existe uma diferença no espaço de busca explorado entre os indivíduos recombinados e mutados, em que os recombinados exploram os locais entre a nova população S^t e os mutados exploram os locais próximos ao $best$ atual. Uma nova iteração começa com uma nova população $S^t + 1$ composta pelo $best$ atual e com os demais $(N - 1)$ indivíduos escolhidos aleatoriamente. Devemos observar que o algoritmo é elitista, ou seja, o *best* encontrado é sempre mantido em uma população.

A recombinação do PSAR emerge de uma instância de uma fórmula de atualização de aprendizagem participativa e se assemelha ao cruzamento aritmético, exceto que é modulado por um grau de compatibilidade e um índice de excitação. A mutação no PSAR também depende do grau de compatibilidade e do índice de excitação da aprendizagem participativa. Isso decorre de um esquema semelhante à mutação da DE (LIU, 2016).

A seguir serão brevemente apresentados e revisados os principais pontos do algoritmo PSAR.

1.2.1 Compatibilidade

Como um fator fundamental no conceito da aprendizagem participativa, que expressa o conhecimento atual e a entrada atual para atualizar o conhecimento, é usado um grau de compatibilidade da seguinte forma:

$$v(t+1) = v(t) + \alpha\rho_t\big(z(t) - v(t)\big), \tag{2}$$

onde $z(t)$ e $v(t)$ são vetores n-dimensionais que indicam o conhecimento atual e a entrada atual, respectivamente. Assumimos, sem perda de generalidade, que $v(t)$ e $z(t)$ $\in [0,1]^n$. O parâmetro $\alpha \in [0,1]$ é a taxa de aprendizagem básica e $\rho_t \in [0,1]$ é o grau de compatibilidade entre $z(t)$ e $v(t)$ no passo *t*. O produto da taxa de aprendizado básico pelo grau de compatibilidade produz a taxa de aprendizado efetiva. Se uma entrada está longe do conhecimento atual, o valor do grau de compatibilidade correspondente é pequeno e a entrada é filtrada, uma vez que a taxa de aprendizado é reduzida pelo grau de compatibilidade. Isso significa que se os dados de entrada estiverem em grande discordância e confusos com o conhecimento atual, eles serão descartados. A maneira adotada para calcular o grau de compatibilidade ρ_t no passo *t* é dada por:

$$\rho_t = 1 - \frac{1}{n}\sum_{k=1}^{n} |z(t) - v(t)| \tag{3}$$

Em um sentido mais geral, ρ_t pode ser visto como uma medida de similaridade entre $z(t)$ e $v(t)$. Se $\rho_t = 0$, então $v(t+1) = v(t)$. Nesse caso $z(t)$ e $v(t)$ são completamente incompatíveis. Essa condição significa que o sistema não está aberto a qualquer aprendizado com as informações atuais. Por outro lado, se $\rho_t = 1$, então $v(t+1) = z(t)$. Já nesse caso, a informação de entrada está totalmente de acordo com os conhecimentos atuais e o sistema está totalmente aberto para aprender.

1.2.2 Seleção

Seja *S* um conjunto de *N* vetores de comprimento fixo *n*, *s* e *s´* $\in S$ sejam dois indivíduos, *s´* distinto de *s*, de modo que:

$$s' = argmax_{r \in S}(\rho(s,r)), \tag{4}$$

onde

$$\rho_t = 1 - \frac{1}{n}\sum_{k=1}^{n} |z(t) - v(t)| \quad (5)$$

e $s = (s_1, s_2, .., s_N)$ e $r = (r_1, r_2, .., r_N)$, sendo que $r \in S$. O indivíduo s´ é aquele cujo índice de compatibilidade com s é o maior. Este procedimento é repetido para cada indivíduo *s* de *S* para montar a população *S´* com *N* indivíduos. Observe que a obtenção dessa população é tendenciosa pelos graus de compatibilidade entre os indivíduos de *S*.

A seleção é feita calculando os graus de compatibilidade entre $s \in S$ e o $s' \in S'$ correspondente com o melhor *best* = s^* atual. A população *L* de indivíduos selecionados é obtida escolhendo o indivíduo de uma das duas populações *S* e *S'* que seja mais compatível com o melhor indivíduo atual. A Figura 2 ilustra o processo de seleção e as seguintes Equações (6), (7) e (8) definem formalmente o melhor s^* e a compatibilidade:

$$\rho^{s'} = 1 - \frac{1}{n}\sum_{k=1}^{n} |s'_k - s_k^*| \quad (6)$$

$$\rho^{s} = 1 - \frac{1}{n}\sum_{k=1}^{n} |s_k - s_k^*| \quad (7)$$

$$s^* = argmin_{s \in S}(f(s, r)) \quad (8)$$

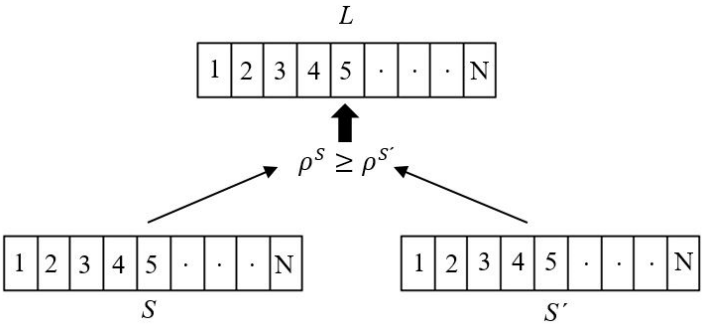

Figura 2 – Seleção no PSAR (Adaptado de LIU, 2016)

A seleção depende da função de aptidão $f(s)$, que identifica o melhor s^* atual, e em ρ^s (s, s^*) e $\rho^{s'}(s', s^*)$ que medem a compatibilidade entre s^* e o par correspondente

de indivíduos e s' da população atual. Em conjunto, f, ρ^{s} e $\rho^{s'}$ decidem se um indivíduo está selecionado ou não (LIU, 2016).

1.2.3 Recombinação

A recombinação deriva da fórmula de atualização de aprendizagem participativa de acordo com a equação:

$$\mathrm{v(t+1)} = \mathrm{v(t)} + \alpha\rho_{\mathrm{t}}^{1-\mathrm{a_t}}\big(\mathrm{z(t)} - \mathrm{v(t)}\big) \tag{9}$$

onde $\mathrm{a_t} \in [0,1]$ é o índice de excitação. A maneira de modelar a excitação é vê-la como o complemento da confiança no conhecimento atual em um procedimento simples, atualizando o índice de excitação a cada etapa t da seguinte forma:

$$a_{t+1} = (1-\beta)\mathrm{a_t} + \beta(1-\rho_{t+1}) \tag{10}$$

onde $\beta \in [0,1]$ controla a taxa de mudança na excitação. Quanto mais alto $\mathrm{a_t}$, menos confiável é o sistema de aprendizado em relação ao conhecimento atual. Se $a_{t+1} = 1$, então temos uma entrada altamente compatível e o índice de excitação diminui. Por outro lado, se $a_{t+1} = 0$, a compatibilidade da informação de entrada é baixa e o índice de excitação aumenta.

Observe que a fórmula da atualização de aprendizagem participativa (9) pode ser reescrita como:

$$\mathrm{v(t+1)} = (1 - \alpha\rho_{\mathrm{t}}^{1-\mathrm{a_t}})\mathrm{v(t)} + (\alpha\rho_{\mathrm{t}}^{1-\mathrm{a_t}})\mathrm{z(t)}. \tag{11}$$

A recombinação participativa prossegue como na equação (9) para produzir o p_r de indivíduos s e s' dos grupos S e S', respectivamente, da seguinte forma:

$$p_r = (1 - \alpha\rho_t^{1-a_t})s + (\alpha\rho_t^{1-a_t})s' \tag{12}$$

O grupo p_r representa os indivíduos resultantes do processo de recombinação.

1.2.4 Mutação

A mutação na BP é semelhante à mutação que ocorre na ED. Na ED uma população de N indivíduos representada por vetores n-dimensionais denotados por $s_{r_i,t}$, na geração t, produz novos indivíduos adicionando as diferenças ponderadas entre os vetores distintos a um terceiro vetor (STORN; PRICE, 1997). Na BP esse processo ocorre produzido o conjunto p_m individual mutado de acordo com a equação (13):

$$p_m = best + \rho_t{}^{1-a}(\ p_L - p_r) \tag{13}$$

O grupo p_m representa os indivíduos resultantes do processo de mutação.

1.2.5 Um Exemplo Ilustrativo do PSAR

Nesta seção o processo de BP é ilustrado usando o PSAR em uma função esfera com duas variáveis. Consideramos para esses tipos de problemas a função esfera a mais simples, definida da seguinte forma:

$$f(s_1, s_2) = (s_1 - 0{,}5)^2 + (\ s_2 - 0{,}5)^2 \tag{14}$$

Onde $s_1, s_2 \in [0{,}1]$. O problema é encontrar s^* que minimize $f(s_1, s_2)$. A função esfera é contínua, convexa e unimodal como mostra a Figura 3, e o mínimo global está em $s^* = (0.5{,}0.5)$ e $f(s^*) = 0$. Assumindo uma população inicial com 3 indivíduos, como mostra a Tabela 1.

Tabela 1 – População inicial

Indivíduo	(s_1, s_2)	$f(s_1, s_2)$
1	(0,0700 0,9000)	0.3449
2	(0,6800 0,7900)	0.1165
3	**(0,5000 0,4100)**	**0.0081**

Da Tabela 1, o indivíduo 3 ($s_1 = 0.6800$ e $s_2 = 0.7900$) é o melhor atual e $f(s_1, s_2) = 0.0081$. Para formar o grupo de reprodução, é encontrado o indivíduo mais compatível entre os indivíduos restantes. Isso é repetido para cada indivíduo da população atual. Por

exemplo, os graus de compatibilidade do indivíduo 1, nomeados por $\rho(1,2)$ e $\rho(1,3)$, são:

$$\rho(1,2) = 1 - \frac{1}{2}(0{,}6100 + 0{,}1100) = 0{,}6400 \quad (15)$$

$$\rho(1,3) = 1 - \frac{1}{2}(0{,}4300 + 0{,}4900) = 0{,}5400 \quad (16)$$

$$\rho(2,3) = 1 - \frac{1}{2}(0{,}1800 + 0{,}38) = 0{,}7200 \quad (17)$$

Tabela 2 – Compatibilidade entre os 4 indivídos

$\rho(s_1, s_2)$	1	2	3
1	-	**0,6400**	0,5400
2	-	-	**0,7200**
3	-	-	-

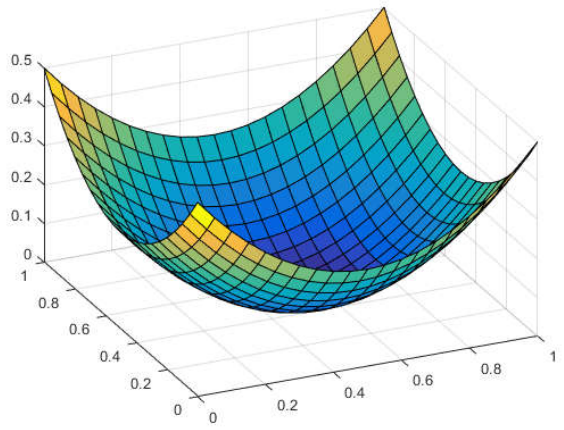

Figura 3 – Função esfera

A Tabela 2 mostra os valores dos graus de compatibilidade para os demais indivíduos da população. Os valores de compatibilidade mais altos, marcados em negrito na Tabela 2, definem os indivíduos mais compatíveis. Os indivíduos de maior compatibilidade formam o conjunto de combinação (S´) como mostrado na Tabela 3 para selecionar os indivíduos mais próximos do melhor indivíduo atual. O algoritmo usa as

equações (6) e (7) e cada par de indivíduos do grupo de reprodução da Tabela 3 para selecionar os indivíduos mais próximos do melhor indivíduo atual. Os indivíduos selecionados (L) são mostrados na primeira coluna da Tabela 4.

Tabela 3 – Conjuntos de combinação

S 1	S´ 2
2	3
3	2

Tabela 4 – Seleção, Recombinação e Mutação

L	p_r	p_m	$f(p_r)$	$f(p_m)$
2	(0,2815 0,8619)	(0,8198 0,4677)	0,1787	0,1033
3	(0,6122 0,6468)	(0,5951 0,6108)	0,0341	0,0213
3	**(0,5678 0,5532)**	**(0,5616 0,5401)**	**0,0074**	**0,0054**

A recombinação é feita para cada par de indivíduos correspondentes do grupo de reprodução atual (S e S´) usando a Equação (12). Os indivíduos recombinados produzidos estão na segunda coluna da Tabela (4). A mutação usa, utilizando a Equação (13), os indivíduos recombinados (p_r) e os selecionados (L) com o melhor indivíduo atual (0,500 0,4100) para produzir a população mutante com indivíduos (p_m). Esses indivíduos são mostrados na terceira coluna da Tabela (4), com os respectivos valores da função objetivo para cada indivíduo mutante coletado na quarta coluna.

Notamos, a partir da Tabela (4), que o valor da função objetivo para o mutado $p_r = (0{,}5616\ \ 0{,}5401)$ é $f(p_r) = 0.0054$ (em negrito) que é melhor que $f(s^*) = 0{,}0081$ do melhor indivíduo atual $(S^*) = (0{,}5000\ \ 0{,}4100)$. Assim, o indivíduo recombinado $p_r = (0{,}5616\ \ 0{,}5401)$ substitui o melhor atual na população da próxima geração.

Os passos da busca participativa detalhada acima estão ilustrados na Figura 4. A Figura 4 mostra o contorno da função objetivo, a população inicial e o melhor mutante e seus respectivos geradores p_r e L. O ponto laranja é a solução ideal $S^* = (0{,}5\ \ 0{,}5)$ que estamos procurando, o roxo é a mutação individual $p_m = (0{,}5616\ \ 0{,}5401)$, o ponto verde é $p_r = (0{,}5678\ \ 0{,}5532)$. Os pontos azuis são a população inicial.

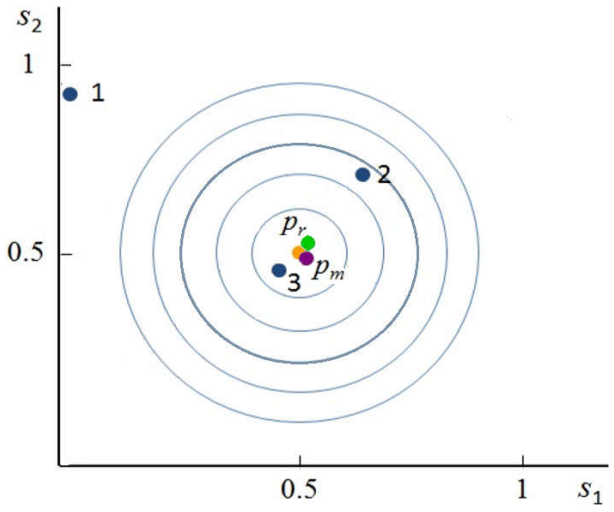

Figura 4 – Amostra de uma etapa do PSAR (Adaptado de LIU, 2016)

1.3 Transferência Seletiva

Nas seções anteriores o estudo do algoritmo PSAR apresentou 3 operadores de pesquisa participativa junto às suas propriedades: a seleção, recombinação e mutação. No trabalho desenvolvido em (LIU, 2016) o operador Transferência Seletiva (TS) é apresentado como mais uma alternativa ao conjunto de operadores participativos. Ele faz parte da sequência de aprendizado das instâncias PSST e PDST. O operador TS se torna significativamente interessante considerando a abordagem combinatória que esse trabalho deseja adotar e que será discutida mais à frente. Dessa forma, esta breve seção pretende apresentar seus conceitos.

A TS é uma substituição filtrada entre indivíduos, sem excluir a possibilidade de que toda a sequência seja copiada (BIRCHENHALL et al., 1997). A TS é semelhante ao crossover clássico apresentado em (HOLLAND, 1975), mas além disso é uma transferência unidirecional entre indivíduos, e não uma troca entre indivíduos. Um exemplo é apresentado na Figura 5. As instâncias PSST e DPST traduzem a ideia desse operador em um procedimento de recombinação da seguinte forma:

Suponha que um indivíduo de L (Pselecionados) seja selecionado usando a função objetiva e a compatibilidade, conforme descrito na Seção 2.2.2. Duas posições $h \leq k$ na sequência do indivíduo são escolhidas aleatoriamente. Com uma probabilidade de 50% as subsequências periféricas a partir de h e k do indivíduo de L são substituídas pelas subsequências correspondentes do indivíduo *best* entre h e k. Caso não seja escolhida a

opção anterior, então a subsequência interna entre *h* e *k* do indivíduo de *L* é substituída pela subsequência correspondente do indivíduo *best* entre *h* e *k*.

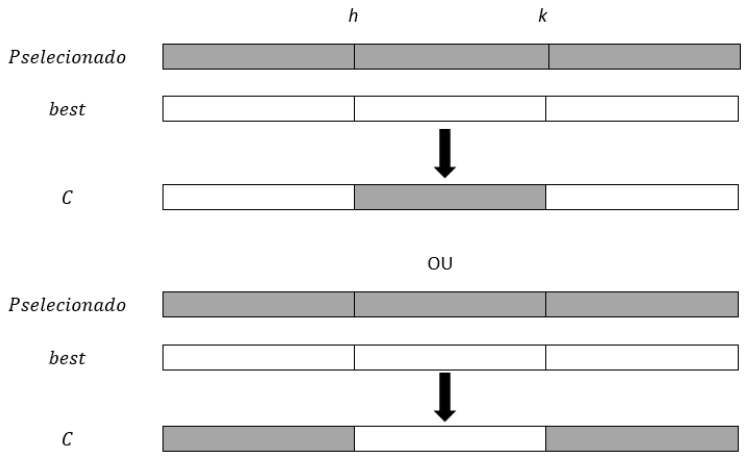

Figura 5 – Transferência Seletiva

2 Árvores de Padrão Fuzzy

Neste capítulo serão apresentados os conceitos básicos e as definições que envolvem o modelo de Árvores de Padrões Fuzzy (APF), além de uma revisão do estado da arte em seus métodos de aprendizado.

2.1 Conceitos básicos

O modelo APF foi introduzido como um método de aprendizado de máquinas para classificação como uma alternativa aos Sistemas Fuzzy Baseado em Regras (SFBR). Muitas estratégias que foram desenvolvidas para induzir modelos de SFBR (CORDÓN, 2011) porém a obtenção de um modelo desse tipo não é uma tarefa simples. No método de APF é concebida uma estrutura hierárquica análoga a uma árvore, onde os nós internos são definidos como operadores lógicos generalizados e as folhas dessas árvores são termos *fuzzy* associados a um atributo de entrada. Uma árvore dessa natureza propaga a informação de baixo para cima. Um nó usa os valores de nós anteriores ou de entradas do sistema como suas entradas, combina-os usando seu respectivo operador e envia a saída para seu predecessor. Uma APF implementa um mapeamento recursivo produzindo saídas no intervalo unitário [0,1]. Um classificador de árvore de padrões consiste em um conjunto de árvores, uma para cada classe. Os valores dos atributos que representam a amostra que se deseja classificar são apresentados nas entradas das árvores de cada classe e a predição da classe é dada pela a árvore que tem o maior valor de saída. As APFs são interessantes por várias razões, especialmente do ponto de vista da interpretação, geralmente cada árvore pode ser considerada como uma "descrição lógica[1]" da classe, permitindo uma interpretação mais concreta do problema de aprendizado permitindo extração de conhecimento do problema analisado (SENGE; HÜLLERMEIER, 2011).

O primeiro método de indução de APF criado por *Huang, Gedeon e Nikravesh* (HUANG; GEDEON; NIKRAVESH, 2008) foi utilizado na área do aprendizado de máquinas em 2008. O algoritmo de geração da árvore foi posteriormente aperfeiçoado em 2011 por *Eike Hullemeier* e *Eyke Hüllermeier* (SENGE; HÜLLERMEIER, 2011). As APFs foram criadas com o intuito de não usar regra para representar o conhecimento. A utilização deste tipo de representação hierárquica procura minimizar os problemas existentes em SFBR,tais como o aumento exponencial do número de regras ao passo que

se aumenta o número de entradas e o comprometimento da interpretabilidade quando uma grande quantidade de regras é gerada para atingir os requisitos de acurácia.

2.2 Folhas

O uso da APF para classificação requer uma base de dados T que pode ser representada pela equação (18), ou seja, n amostras $x \subset X$ sendo cada uma delas possuidora de rótulo $y \subset Y$ que indica a classe à qual a amostra pertence. As entradas da APF podem ser chamadas também de folhas, representado um nível mais baixo onde os atributos particionados em conjuntos *fuzzy* são inseridos.

$$T = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n} \subset X \times Y \tag{18}$$

Na equação (18) cada instância $x^{(i)}$ é um vetor associado a um rótulo $y^{(i)}$ de sua classe, onde:

$$x \in X = X_1 \times X_2 \times \ldots \times X_m \tag{19}$$

$$y \in Y = \{y_{1,} y_{2,\ldots,} y_k\} \tag{20}$$

X_i é o domínio do atributo i-ésimo atributo A_i. Cada domínio é particionado em um conjunto de funções de pertinência F_{ij}

$$F_{i,j}: X_i \to [0,1] \quad (j = 1, \ldots, n_i) \tag{21}$$

De tal modo que $\sum_{j=1}^{n_i} F_{i,j}(x) > 0, \forall\, x \in X_i$. $F_{i,j}$ normalmente recebe um rótulo linguístico como, por exemplo, “ALTO” ou “BAIXO”. Neste caso, ele é chamado de termo linguístico.

Ao contrário das Árvores de Decisão (BREIMAN, 2001), a entrada da APF está em suas folhas. Cada folha da árvore é rotulada por um atributo A_i e um conjunto *fuzzy* $F_{i,j}$ do domínio X_i correspondente. Dada uma instância $x = (x_{1,} x_{2,\ldots,} x_m)$ como entrada, o nó folha produz, como saída $F_{i,j}(x_i)$, que é o grau de pertinência de x_i em $F_{i,j}$. Este

grau é propagado para outros nós em direção a raiz. Assim como uma árvore invertida, essa raiz fica localizada no topo.

2.3 Operadores

São utilizados operadores *fuzzy* na criação das árvores, sendo os utilizados nesta dissertação os seguintes: *t-norm* e *t-conorm* (KLEMENT; MESIAR; PAP, 2013) e dois operadores de média, *Weighted Average* (WA) e *Ordered Weighted Average* (OWA). O operador de média OWA é a combinação de k números, $v_{1,}v_{2,\dots,}v_k$, definido pela equação (22).

$$OWA_w(v_{1,}v_{2,\dots,}v_k) \stackrel{\text{def}}{=} \sum_{i=1}^{k} w_i v_t(i) \tag{22}$$

Onde o τ é uma permutação entre o conjunto de números $\{1,2,\dots,k\}$ de forma que $v_{t(1)} \leq v_{t(2)} \leq \cdots \leq v_{t(k)}$ e $w = (w_{1,}w_{2,\dots,}w_k)$ é um vetor de pesos que satisfaz $w_i \geq 0$ para $i = 1,2,\dots,k$ e o somatório dos pesos de 1 a k deverá ser igual a 1. O operador de média WA (*Weighted Average*) é similar ao OWA, porém sem o ranqueamento dos valores de *v*.

Os detalhes dos demais operadores (*t-norm* e *t-conorm*) utilizados na construção das APFs são apresentados nas Tabela 5 e Tabela 6.

Tabela 5 - Operadores Fuzzy T-Norm

Mínimo	$\min(a,b)$
Algébrico	$a * \mathrm{b}$
Lukasiewicz	$\max(a-1+b,0)$
Einstein	$\dfrac{a * \mathrm{b}}{2-(a+b-a * \mathrm{b})}$

Tabela 6 - Operadores Fuzzy T-Conorm

Máximo	$\max(a,b)$
Algébrico	$a+b-a * \mathrm{b}$
Lukasiewicz	$\min(a+b,1)$
Einstein	$\dfrac{a+\ \mathrm{b}}{1+(a * \mathrm{b})}$

2.4 Exemplo de estrutura

Um exemplo de APF pode ser visto na Figura 6. No exemplo a árvore representada determina uma classe que representa a qualidade de um vinho, tendo como possíveis atributos de entrada o teor alcoólico, a acidez, a concentração de sulfatos e de dióxido de enxofre. Estes atributos estão associados a um termo *fuzzy* que representa um intervalo do universo de discurso do atributo, transformando os valores dos atributos antes da inserção na árvore em graus de pertinência. O valor de pertinência obtido nos conjuntos *fuzzy* são inseridos nas folhas e se agrupam através dos operadores de forma que esses resultados parciais contidos no decorrer da árvore se mantêm no intervalo [0,1]. O valor obtido na saída após todos os agrupamentos das características deve se aproximar de 1 se a árvore representa bem a classe. No exemplo, um vinho com boa qualidade terá na saída da árvore um valor próximo de 1.

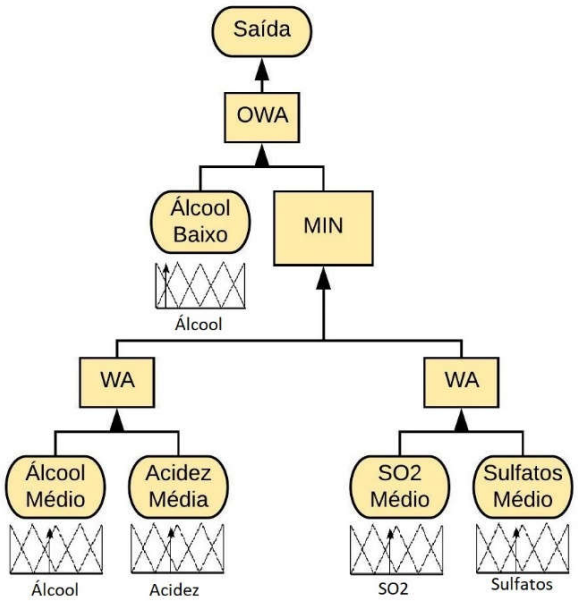

Figura 6 – APF para classificar a qualidade de vinhos

Através do exemplo da Figura 6 é notável que a qualidade do vinho está relacionada com duas subárvores. A primeira relaciona o teor alcoólico e a acidez e a segunda subárvore trata da concentração de sulfato e de dióxido de enxofre (SO2), sendo que a estas duas primeiras árvores são combinadas em um nível mais alto. No nível acima, o resultado da avaliação das duas subárvores é combinado com o nível alcoólico tendo este uma grande influência no resultado pois está localizado próximo ao topo da árvore.

A combinação em todos os níveis da árvore é realizada utilizando os operadores contidos na sua construção. Neste sistema é possível avaliar que as subárvores geradas representam diferentes conhecimentos que devem ser combinados, além da relação que existe entre as variáveis.

Para o caso multiclasse, um classificador APF pode ser visto como uma coleção de APF dada por:

$$\{APF_i | i = 1,2, \dots, k\}$$

Onde a APF_i é a árvore associada a classe com a classe y_i. Dada uma nova instância x para ser classificada, a decisão $\hat{y}$ é tomada em favor da classe cuja a árvore apresenta o maior valor na saída:

$$\hat{y} = argmax_{y_i \in Y} APF_i(x)$$

O modelo baseado em árvore de padrões mapeia diversas entradas fuzzificadas a partir das características de entrada, em apenas uma variável de saída, sendo gerada uma variável de saída para cada classe que a base de dados possuir. A interpretação da saída produzida permite a análise das características mais relevantes da base de dados em análise. Essa análise permite avaliar a importância de subcritérios de cada característica.

2.5 Métodos de aprendizado

2.5.1 Bottom-up Induction

O primeiro método síntese das APF foi o *Bottom-Up Induction* criado por Huang, Gedeon e Nikravesh (HUANG; GEDEON; NIKRAVESH, 2008) e aperfeiçoado posteriormente por Eike Hullermeier (SENGE; HULLERMEIER, 2011). Nele, a indução das árvores procura criar uma representação lógica para cada classe de uma forma iterativa. A descrição lógica pode ser considerada não inteiramente lógica uma vez que se torna permitido o uso de operadores aritméticos. O processo se propaga como o nome do modelo sugere, da base para o topo. Em cada iteração, combinam-se as duas melhores árvores candidatas para criar uma árvore. Ela pode ser vista como uma combinação iterativa para a construção de características complexas, a partir de características mais básicas criadas pelos atributos originais (SENGE; HÜLLERMEIER, 2011).

A combinação de duas árvores candidatas para gerar uma terceira tende a realizar "grandes saltos" no espaço de busca e a gerar uma perda de diversidade, porque após algumas interações todas as árvores candidatas tornam-se semelhantes (SANTOS; AMARAL, 2014).

2.5.2 Top-down Induction

Nesta estratégia, ao invés de juntar duas árvores em uma nova árvore maior e com certa diferença na estrutura, a ideia é fazer pequenas modificações. Isto é feito expandindo um nó de folha de cada vez. Na Figura 7 é possível visualizar uma sequência de criação da árvore formada apenas a partir da folha 1A, que representa a junção de um atributo e um termo *fuzzy*. Ela é substituída por uma característica composta pela mesma folha 1A e outra folha 2B, agregadas por um operador (Op.1). Novos operadores são inseridos em níveis mais baixos da árvore, mais ao fundo, isto faz com que cada novo operador tenha uma influência menor no comportamento entrada-saída que os operadores inseridos anteriormente. Este procedimento de realizar pequenas modificações na árvore a cada iteração proporciona uma maior exploração do espaço de busca, o que aumenta a chance de encontrar uma árvore que atenda aos requisitos do usuário.

O processo da criação das árvores, inicia-se com uma árvore primitiva que é um subconjunto *fuzzy* no domínio do atributo. Ocorre então a expansão das árvores candidatas de forma iterativa selecionando a melhor árvore baseando-se em um critério de medida, até que o critério de término seja atingido.

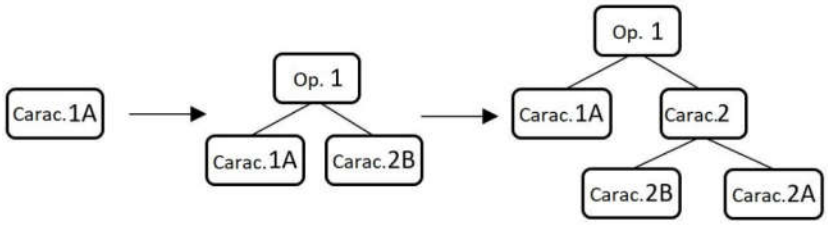

Figura 7 – Exemplo da sequência de síntese de APFs no método Top-down Induction

No processo *Top-down Induction* de síntese, descrito a cima, utiliza-se o algoritmo *Beam Search*. Esse algoritmo apresenta uma estratégia do tipo *best first search*. Ele explora a árvore expandindo o nó mais promissor a partir de um conjunto limitado de opções, nesse mecanismo, a cada nível da árvore gera-se todos possíveis sucessores dos

nós do nível atual, organizando-os em uma ordem crescente de custo heurístico. Porém, para reduzir os requisitos de memória, este método só armazena um número pré-determinado (largura do feixe de busca) de melhores sucessores de cada nível. Somente estes melhores sucessores serão expandidos futuramente. O método *Top-down Induction* apresenta algumas deficiências, tais como: Quantidade de atributos pode se tornar excessiva com o aumento da largura do feixe, gerando dificuldades de processamento. Em contrapartida, uma largura de feixe pequena gera problemas para explorar todo o espaço de busca; O algoritmo *Beam Search* pode ficar preso em sub ótimos globais pois está sempre direcionado em grande parte para o melhor candidato no estágio de construção; O critério de parada utilizado não põe limites ao crescimento da árvore o que pode gerar *overfitting* além de árvores menos interpretáveis.

2.5.3 Programação Genética Cartesiana

Em (SANTOS, 2014) foi desenvolvido um método de síntese de APF a partir da Programação Genética Cartesiana (PGC) (MILLER; THOMSON, 2000). Nele se utiliza do artifício da representação em grafos que a PGC possui. O grafo é codificado em uma sequência linear de inteiros e são representados em uma grade de nós computacionais de n-dimensões, com a extensão definida pelo programador, mas que, em geral possui uma ou duas dimensões. Na Figura 8 é dado um exemplo da sequência linear de inteiros que é denominada cromossomo ou genótipo.

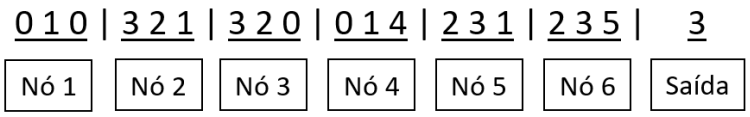

Figura 8 – Genótipo ou cromossomo na PGC

A junção de um gene de função com alguns genes de conexão forma um nó. A Estrutura de cada nó consiste em uma função (gene de função) em que são aplicados os valores de entrada cuja localização é dada pelo gene de conexão, para gerar uma saída, a função é escolhida dentro de um conjunto de funções previamente definidas de acordo com a aplicação. A entrada de cada nó pode ser uma entrada do sistema ou a saída de um nó anterior. É chamado de *levelback*, o máximo de nós anteriores que um gene de conexão pode se conectar. A quantidade de entradas dos nós é chamada de aridade, e é determinada

de acordo com a função que necessita do maior número de entradas entre as funções do conjunto. Tanto o *levelback* como a aridade, são parâmetros a serem definidos pelo programador (MILLER; HARDING, 2009; MILLER; THOMSON, 2000; MILLER, 2011).

A estratégia evolutiva utilizada na PGC foi $1 + \lambda$ em que $\lambda = 4$ podendo ser descrito nos seguintes passos: Cinco Indivíduos (genótipos) são criados aleatoriamente. O indivíduo que possui o maior valor da função de aptidão será considerado o mais apto e será promovido para a população da geração seguinte. O operador mutação é aplicado quatro vezes no indivíduo mais apto da geração anterior, gerando quatro novos indivíduos, totalizando cinco indivíduos na população atual. É escolhido novamente o indivíduo mais apto. Caso haja mais de um indivíduo com a melhor aptidão, sempre será promovido para a próxima geração aquele indivíduo que tenha sido criado na geração atual. O processo anterior é repetido até que um critério de parada seja atingido (SANTOS, 2014).

A representação utilizada na PGC pode ser facilmente utilizada para representar APF. Além disso, a PGC é um algoritmo de busca global capaz de explorar grandes espaços de forma eficiente. A exploração do espaço de busca na PGC não segue a estratégia gulosa do *Beam Search*, portanto tem mais chances de obter soluções melhores. Além disso, a exploração do espaço não é feita de forma restrita como no caso da *Beam Search* (que é limitada pela largura do feixe), o que também aumenta as chances de obter soluções melhores. A utilização da PGC também facilita a utilização de operadores parametrizados, pois o operador de mutação é capaz de alterar os parâmetros destes operadores com o objetivo de melhorar o desempenho (SANTOS, 2014).

O método de indução de APFs através de PGC se mostrou com desempenho similar aos demais algoritmos propostos por (SENGE; HÜLLERMEIER, 2011) e (HUANG; GEDEON; NIKRAVESH, 2008). Além de apresentar um número inferior de avaliações durante o processo de aprendizagem ele gerou árvores mais compactas. As vantagens do método advém de não sofrer tanto com a "maldição da dimensionalidade" e realizar uma busca de forma mais global.

3 Algoritmos Evolucionários para Problemas Combinatórios

Neste trabalho, utilizou-se a mesma representação para a síntese das APF que foi utilizada por (SANTOS, 2014), que descreve o grafo que representa uma APF através de uma sequência linear de inteiros com restrições. Quando se utiliza esta representação, o problema de encontrar uma APF, que represente um classificador, pode ser visto como um problema de otimização combinatória, em que se deseja maximizar ou minimizar uma função de avaliação dentro de um espaço de busca.

Existem diversas meta-heurísticas consolidadas que definem estratégias evolucionárias que "guiam" o processo de busca, tais como: Algoritmo Genéticos (HOLLAND, 1975), Estratégias Evolutivas (RECHENBERG, 1973) (SCHWEFEL, 1975; 1977), Programação Evolutiva (FOGEL et al, 1966) ou a Programação Genética (KOZA, 1992). Todos esses métodos são modelos de busca que apresentam diferenças em relação às possibilidades de representação, auto-adaptação, à função de avaliação e à utilização de operadores de seleção, recombinação e mutação. Este capítulo pretende abordar as principais questões envolvidas nos problemas de otimização combinatória e as possíveis abordagens e alternativas no tratamento desse problema dentro da síntese das APF. Nas seções que compõem este capítulo, uma breve revisão bibliográfica dos algoritmos evolucionários e seus operadores será apresentada.

3.1 Problemas de Otimização Combinatórios

Nesta seção serão apresentados alguns conceitos e definições para melhor entendimento do problema que se deseja tratar, assim como sua formulação matemática.

Problemas que envolvem otimização são onipresentes em toda a comunidade científica sendo de grande importancia sua abordagem. Segundo (BLUM; ROLI, 2008), um problema *P* de otimização pode ser descrito como sendo composto por três principais elementos: S, Ω e f, onde:

- S é o espaço de busca definido sobre um conjunto finito de variáveis de decisão $X_i, i = 1, \ldots, n$. Caso essas variáveis tenham domínios discretos, *P* é chamado problema de otimização combinatória (ou problema de otimização discreto),

e em casos de domínios contínuos trata-se de um problema de otimização contínua. Problemas com variáveis mistas também existem.

- Ω é um conjunto de restrições entre as variáveis;
- $f: S \rightarrow \mathrm{R}^{+}$ é a função objetivo que especifica um valor positivo para cada elemento (ou solução) de *S*.

O objetivo de um problema de otimização é encontrar a melhor solução $s^* \in S$ tal que $f(s^*) \leq f(s), \forall\, s \in S$ para um problema de minimização, ou $s^* \in S$ tal que $f(s^*) \geq f(s), \forall\, s \in S$ para um problema de maximização. Em diversos problemas, o objetivo é frequentemente otimizar várias funções objetivos ao mesmo tempo. Essa forma de otimização é definida como otimização multiobjetivo.

No processo de otimização existem dois principais pontos a considerar no espaço de busca definido, o ótimo local e global. O ótimo local $\hat{s} \in S$ de uma função f é um elemento $f(\hat{s}) \geq f(s')$ para todos os s' vizinhos a $\hat{s}$, ou seja, não é possível por meio de um movimento discreto encontrar uma solução vizinha melhor que a solução corrente. O ótimo global refere-se ao maior valor da função objetivo, entre todos os ótimos locais existentes no espaço de busca. A função objetivo dos problemas de otimização são frequentemente multimodais, ou seja, existem múltiplos ótimos locais e globais. A existência de múltiplos ótimos locais dificulta consideravelmente a busca pelo ótimo global.

A Otimização Combinatória (OC) trata problemas de maximização ou minimização de uma função de uma ou mais variáveis, sujeitos a restrições de igualdade ou desigualdade e restrições de integralidade de algumas ou de todas as variáveis (NEMHAUSER; WOLSEY, 1988). Em (PAPADIMITRIOU; STEIGLITZ, 1982) problemas de OC são considerados à busca pela solução de um objeto que é geralmente um número inteiro, um subconjunto, uma permutação, ou a estrutura de um grafo. Esse objeto é comumente chamado de solução do problema e pode ser considerado um ótimo global ou ótimo local dependendo da posição que essa solução se encontre no espaço de busca.

Uma das principais características que diferem os problemas de OC dos demais problemas de otimização é em relação ao domínio das variáveis. O domínio de uma variável em um problema de OC representa o conjunto de possíveis valores discretos que podem ser atribuídos a esta variável. Além disso, pode haver restrições em relação às

variáveis. No caso de problemas de permutação, como o Problema do Caixeiro Viajante (*Traveling Salesman Problem* - TSP) (LAWLER ,1985) por exemplo, nenhum valor pode se repetir, ou seja, neste tipo de problema um mesmo valor não pode ser atribuído a duas ou mais variáveis de uma mesma solução (ROSENDO, 2010).

Considerando que se deseja minimizar $f(x)$ e incorporar restrições à função objetivo, podemos formular problemas de OC da seguinte forma:

$$\begin{aligned} &\boldsymbol{x} \in X = \{(x_1, x_2, \dots, x_n) \mid x_i \in U, \forall i = 1,2, \dots, n \; e \; U \text{ finito}\}, \\ &\boldsymbol{g}(\boldsymbol{x}) \leq 0, \end{aligned} \tag{23}$$

onde: $\boldsymbol{g}: X \to \mathbb{R}^q, \boldsymbol{x} \to \boldsymbol{g}(\boldsymbol{x})$ e $\boldsymbol{f}: X \to \mathrm{R}, \boldsymbol{x} \to \boldsymbol{f}(\boldsymbol{x})$

Considerando que $-\infty < f(\boldsymbol{x}) < +\infty, \forall\, \boldsymbol{x} \in X$, então se $c(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{w}\boldsymbol{h}(\boldsymbol{x})$, onde $\boldsymbol{w} = (\mathrm{M}, \mathrm{M}, \dots, \mathrm{M}) \in \mathbb{R}^q$, sendo M um número arbitrariamente grande, e:

$$h_i(\boldsymbol{x}) = \begin{cases} 0, & se \quad g_i(\boldsymbol{x}) \leq 0 \\ 1, & caso \; conttrário \end{cases} \tag{24}$$

A restrição $\boldsymbol{g}(\boldsymbol{x})$ pode ser incorporada na função objetivo e considerando minimizar $\boldsymbol{g}(\boldsymbol{x})$, a equação (23) passa a ser:

$$\boldsymbol{x} \in X = \{(x_1, x_2, \dots, x_n), \text{tal que } x_i \in U, \forall i = 1,2, \dots, n \; e \; U \text{ finito}\}, \tag{25}$$

onde: $\boldsymbol{f}: X \to \mathrm{R}, \boldsymbol{x} \to \boldsymbol{f}(\boldsymbol{x})$

Outra formulação possível é considerar uma função *h* definida por:

$$h(\boldsymbol{x}) = \boldsymbol{v}\boldsymbol{g}(\boldsymbol{x}), \text{ onde } \boldsymbol{v} \in \mathbb{R}^q \text{ e } \boldsymbol{v}_i(\boldsymbol{x}) > 0 \text{ se } \boldsymbol{g}_i(\boldsymbol{x}) < 0 \text{ e } \boldsymbol{v}_i = 0 \text{ se } \boldsymbol{g}_i(\boldsymbol{x}) \geq 0, \tag{26}$$

e utilizar uma função *c*, expressa por $c(x) = f(x) + h(x)$. A diferença entre as funções *h* expressas por (24) e (26) ocorre do fato que em (24) uma solução $\boldsymbol{x}$ que não satisfaça $\boldsymbol{g}(\boldsymbol{x}) \leq 0$ será penalizada com um valor suficientemente alto (M), ao passo que em (26) a penalização na função objetivo será proporcional a quanto que a restrição $\boldsymbol{g}(\boldsymbol{x}) \leq 0$ é violada. Desse modo, as desigualdades $\boldsymbol{g}(\boldsymbol{x}) \leq 0$ não serão mais tratadas como restrições (ou seja, soluções $\boldsymbol{x}$ que apresentam $\boldsymbol{g}(\boldsymbol{x}) > 0$ serão consideradas válidas), mas induzirão

custos na função objetivo. As restrições podem estabelecer regras mais específicas como, por exemplo: num determinado problema nenhuma solução pode ter o valor z atribuído a uma variável precedente de outra variável com o valor y, assim, toda solução que tivesse a estrutura $s = \{\dots, (x_i, z), (x_{i+1}, y), \dots\}$ seria considerada uma solução não viável para este problema.

3.2 Algoritmos Evolucionários em Problemas Combinatórios

Nesta seção será feita uma breve revisão bibliográfica dos Algoritmos Evolucionários (AE) no tratamento de problemas que envolvem otimização discreta assim como os operadores genéticos que se relacionam com este tipo de problema.

No ambiente natural, a evolução se estabelece como um mecanismo capaz de criar organismos autônomos e complexos com habilidades de sobreviverem em ambientes imprecisos e em constante transformação. Desse modo, na natureza, o processo de evolução se manifesta como um processo de adaptação robusto, onde indivíduos superiores, são equipados com diversas habilidades sendo bem-sucedidos na busca pela sobrevivência em diferentes ecossistemas (DRÉO, 2006). Sendo a evolução o resultado da reprodução, variação genética e seleção natural aplicados à uma população de indivíduos.

Nas décadas de 60 e 70, três vertentes surgiram relacionando a evolução biológica como uma ferramenta de otimização (MELANIE, 1998), sendo elas: as Estratégias Evolucionárias por (RECHENBERG, 1965); a Programação Evolucionária por (FOGEL; OWENS; WALSH, 1966); e os Algoritmos Genéticos por (HOLLAND, 1975). Todas essas três vertentes apresentavam características semelhantes envolvendo reprodução, comportamento aleatório, competição e seleção de indivíduos pertencentes à uma determinada população (FOGEL, 1995).

Na década de 90, no evento de pesquisadores de algoritmos evolutivos na conferência *Parallel Problem Solving from Natur*e (PPSN) realizada em Dortmund e na conferência *International Conference on Genetic Algorithms* (ICGA'91) realizada em 1991, em um esforço para reunir as diversas abordagens, que já apontavam para uma evidente necessidade de agrupamento devidos à constante interação entre elas, surge a área de pesquisa avançada da Computação Evolutiva (CE), que trata os Algoritmos Evolucionários (AE). Nessa época o Algoritmo Genético (AG) se consolidava como a

técnica mais proeminente da CE (DAVIS, 1991). Em 1994 a CE é incluída na primeira *World Conference on Computational Intelligence* (WCCI).

Dentro dos AE existem variadas técnicas que se ramificaram e evoluíram no decorrer dos anos. Atualmente, os AE são utilizados nos mais diversos problemas como, por exemplo, classificação, design de circuitos, rotas e otimizações em geral (OLIVEIRA, 2017).

Existem diversos AE, tais como: Algoritmo Genéticos (HOLLAND, 1975), Estratégias Evolutivas (RECHENBERG, 1973) (SCHWEFEL, 1975; 1977), Programação Evolutiva (FOGEL et al, 1966) ou a Programação Genética (KOZA, 1992) que surgiu mais tarde e também se estabelece como uma importante metaheurística. Eles se consolidam como vertentes originais dentro da área de CE, porém é importante notar que todos eles são passíveis de variações e hibridizações.

O desempenho dos operadores de pesquisa varia entre os diferentes estágios do processo de busca dos AE. Em geral, um único operador de pesquisa pode não se sair bem em todos esses estágios quando lida com diferentes problemas de otimização e busca. Para atenuar isso, esquemas de operadores de busca adaptativos também são desenvolvidos e aprimorados em algumas pesquisas. Em (MASHWANI; et al, 2017), por exemplo, é utilizado um AE adaptativo híbrido que avalia e altera o operador de variação genética de acordo com a eficiência do processo evolutivo. O estudo gerou resultados promissores que se equiparam à vários algoritmos bem conhecidos e consolidados.

Existe uma fronteira que a cada dia se torna mais tênue entre os AE devido às variações e hibridizações que ocorrem. Devido à essa tendência, as pesquisas na área da CE preferem em muitos casos descrever seus algoritmos como algoritmos evolutivos com características específicas ao invés de categorizá-los como um AE específico.

3.2.1 Algoritmo Genéticos

Os AG foram desenvolvidos por John H. Holland em (HOLLAND, 1975) e (HOLLAND, 1992) para a resolução de problemas em diversas áreas, tendo como metas abstrair e explicar rigorosamente os processos adaptativos em sistemas naturais de forma não determinística. Segundo (ROZENBERG; BCK; KOK, 2011), os AG são algoritmos que dão ênfase ao uso de genótipo como forma de representação das soluções candidatas, sendo decodificados e avaliados a cada interação do algoritmo. Os genótipos são apresentados em estruturas de dados simples e são definidos como cromossomos

artificiais. Em boa parte das aplicações envolvendo AG, os genótipos são formados por cadeias genes que são recombinadas em uma forma simplificada de reprodução sexuada e podem sofrer mutações com trocas aleatórias de genes dentro da cadeia ao longo das gerações. Os operadores genéticos devem ser definidos considerando a forma de representação adotada e a natureza do problema em questão.

O AG é uma das meta-heurísticas eficientes e aplicáveis a um grande conjunto de problemas combinatórios (VARUN KUMAR; PANNEERSELVAM, 2017). Diferente de outros algoritmos que precederam o AG, ele não se comporta apenas como uma estratégia de busca local baseada no conceito de mutações, os AG incorporaram a noção de recombinação (LUZIA; RODRIGUES, 2009). Os conceitos de mutação e recombinação são de grande importância e são utilizadas em diferentes processos evolutivos. Essas operações modificam indivíduos e combinam soluções parciais, assim como na mutação e recombinação dentro da biológica. O cruzamento promove o processo de memória genética à evolução. A mutação por sua vez permite o processo de exploração, evitando que a busca permaneça em mínimos locais (OLIVEIRA, 2017).

Além do próprio AG, os operadores de recombinação e mutação fazem parte de uma grande parcela dos AE o que torna importante um estudo das técnicas disponíveis, uma vez que existe uma interdependência entre a representação empregada e os operadores genéticos escolhidos. Nas seções 3.2.1.1 e 3.2.1.2, apresentadas a seguir, serão identificados os principais tipos de operadores genéticos de recombinação e mutação a serem empregados dentro de uma representação combinatória próxima a que este trabalho pretende utilizar.

3.2.1.1 Operadores de Recombinação

A recombinação é um dos operadores proeminentes usados em AG. O processo de cruzamento é vital para gerar novos cromossomos ao combinar dois ou mais cromossomos parentais com a esperança de que eles criem novos e eficientes cromossomos. Nesse processo não se produz material genético novo, apenas recombina o já existente. O processo ocorre após a seleção dos cromossomos progenitores e a troca de informações entre eles para criar descendentes. Durante o cruzamento, os cromossomos progenitores são trocados em determinada ordem para obter seus

descendentes. Estes descendentes tornam-se cromossomos progenitores da próxima geração (VARUN KUMAR; PANNEERSELVAM, 2017).

Existe diversas possibilidades de operadores de recombinação disponíveis na literatura. Deve-se avaliar a utilização de operadores que favoreçam as novas características de uma representação específica afim de que indivíduos ainda sejam válidos na reavaliação de acordo com os critérios estabelecidos, e de que a recombinação cumpra seu papel de memória genética no processo evolutivo de maneira eficiente. A eficiência do operador utilizado varia de acordo com o tipo de problema a ser tratado e a representação cromossômica adotada para o problema. Em (SILVA, 2017) por exemplo, um estudo foi realizado com o objetivo de determinar a melhor combinação de operadores de seleção, recombinação e mutação no problema do Caxeiro Viajante (*Traveling Salesman Problem* - TSP) (PAPADIMITRIOU,1977) utilizando o AG. O estudo demonstrou experimentalmente que a combinação dos operadores Torneio (Seleção), CX (Recombinação) e Inversão (Mutação) foram os mais eficientes nesse caso.

Cada operador de recombinação é particularmente eficiente para uma determinada classe de problemas e extremamente ineficiente para outras, como também um único operador pode ser aplicável em diferentes classes de problemas e representações. A seguir será apresentada uma coletânea de alguns operadores de recombinação conhecidos na literatura aplicáveis em problemas de otimização discretos afim de inspirar possibilidades de uso no problema proposto.

Os operadores de recombinação apresentados nesta revisão da literatura envolvem as representações binária e de permutações, e servem de inspiração para o desenvolvimento de novos operadores envolvendo a representação cromossômica utilizada neste trabalho. Em (ALGETHAMI, 2015) é realizado um estudo entre diversos operadores para um problema de otimização discreto, e usaremos a mesma distinção entre dois grupos de operadores. Dividiremos os operadores de recombinação apresentados nos grupos: *Scheduling Operators* e *Routing Operators*. O grupo de operadores *Scheduling Operators* foram desenvolvidos originalmente para uma representação binária, o grupo *Routing Operators* foram desenvolvidos originalmente para representação de permutações de números inteiros sem repetições, aplicado em problemas de ordem como o TSP. A Tabela 7 apresenta um resumo dos operadores que serão abordados nesta revisão.

Tabela 7 – Operadores de recombinação considerados no estudo e adotados na literatura

Operador	Nome do Operador	Nomenclatura	Referência
SchedulingOperators	Crossover de 1 ou *k*pontos (*K-Point Crossover*)	1PX 2PX KPX	(HOLLAND, 1975) (MITCHELL,1998) (HARTMANN, 1998)
	Uniform Crossover	UX	(MITCHELL,1998)
	Half-uniform Crossover	HX	(ESHELMAN, 1991)
RoutingOperators	*OrderCrossover*	OX1	(DAVIS, 1985)
	OrderBased Crossover	OX2	(SYSWERDA, 1991)
	CycleCrossover	CX	(OLIVER; et al, 1987)
	PartiallyMapped Crossover	PMX	(GOLDBERG; et al, 1985)
	ModifiedPartially-Mapped Crossover	MPMX	(OLIVER; et al, 1987)
	Maximal PreservationCrossover	MPX	(SYSWERDA, 1991)
	PositionBased Crossover	POS	(MÜHLENBEIN; et al, 1988)
	VotingRecombination Crossover	VR	(MÜHLENBEIN, 1989)

- Recombinação de 1 ou *k* pontos (1PX – 2PX – KPX):

É uma das técnicas de recombinação mais simples usadas para diversos tipos de representação em AG. Dado um par de indivíduos (cromossomos), primeiro é definido um ponto de corte, no qual a proporção pode se tornar um parâmetro do operador, sendo aleatório ou definido previamente. Posteriormente, a primeira parte do primeiro cromossomo será agrupada à segunda parte do segundo cromossomo, formando assim o

primeiro cromossomo descendente, depois, a primeira parte do segundo cromossomo será agrupada à segunda parte do primeiro cromossomo formando o segundo cromossomo descendente (DRÉO,2006).

Seja, como exemplo, um par de progenitores, da Figura 9, cada um com cromossomo de 9 genes. Assim, é dada ums posição de corte entre os genes 4 e 5. Para apreciação visual o progenitor 1 está grafado em negrito enquanto o progenitor 2 não.

Progenitor 1:	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**
Progenitor 2:	1	1	0	1	1	0	1	1	0
Descendente 1:	**1**	**0**	**1**	**1**	1	0	1	1	0
Descendente 2:	1	1	0	1	**0**	**0**	**0**	**1**	**1**

Figura 9 – Recombinação de 1 ponto

Na recombinação de 2 ou *k*-pontos de cortes, são definidos os múltiplos pontos de corte da mesma maneira como no operador de 1 ponto. As características genéticas contidas entre os pontos de corte são trocadas alternadamente entres os cromossomos progenitores. A Figura 10 apresenta um exemplo desse tipo de recombinação, onde o progenitor 1 tem seus genes em negrito e 3 pontos de cortes são definidos entre os genes 2 e 3, entre os genes 4 e 5 e entre os genes 7 e 8. O material genético é então trocado entre os progenitores a partir dos pontos de corte, gerando os descendentes.

Progenitor 1:	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**
Progenitor 2:	1	1	0	1	1	0	1	1	0
Descendente 1:	**1**	**0**	0	1	**0**	**0**	**0**	1	0
Descendente 2:	1	1	**1**	**1**	1	0	1	**1**	**1**

Figura 10 – Recombinação de *k*-pontos

- *Uniform Crossover*

Para o crossover uniforme gera-se, aleatoriamente, uma máscara constituída de "*true*" = "1" e "*false*" = "0" em quantidade igual à quantidade de genes em cada cromossomo, correspondentes às posições dos respectivos genes dos cromossomos. A geração aleatória dos "1" ocorre com probabilidade $p_u, 0 < p_u \leq 0$ e a geração dos "0" ocorre com probabilidade $(1 - p_u)$. Onde p_u se torna um parâmetro do operador.

Após a geração da máscara realiza-se então a recombinação dos genes dos progenitores. Inspeciona-se então a máscara de "0" e "1" e para as posições nas quais o valor é "1" troca-se os genes correspondentes dos cromossomos dos progenitores para aquela posição. Na Figura 11 é apresentado um exemplo entre dois progenitores e uma máscara previamente gerada. Em negrito estão marcados os valores iguais a "0", os quais definem as posições onde não serão trocados os genes dos cromossomos, ou seja, onde existe o valor "0" na máscara, o descendente 1 recebe o gene do progenitor 1, onde existe o valor "1" na máscara, o descendente 1 recebe o gene do progenitor 2. Ocorrendo o mesmo para o descendente 2.

Máscara:	**0**	**0**	**0**	1	1	1	**0**	1	**0**
Progenitor 1:	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**
Progenitor 2:	1	1	0	1	1	0	1	1	0
Descendente 1:	**1**	**0**	**1**	1	1	0	**0**	1	**1**
Descendente 2:	1	1	0	**1**	**0**	**0**	1	**1**	0

Figura 11 – Recombinação Uniforme

- *Half-uniform Crossover*

No operador *Half-uniform Crossover*, os valores em cada posição do par selecionado de progenitores são comparados. Se o valor do gene em um determinado local é igual em ambos cromossomos, ele é mantido intacto. Se o gene diferir, um novo

valor será retirado de um dos cromossomos progenitores com probabilidade igual. Isto é ilustrado na Figura 12 com um exemplo. Nele as posições 2, 4, 7 e 8 apresentam valores iguais entre progenitores, os quais serão mantidos em ambos descendentes. Nas outras posições ocorre o sorteio onde se escolhe de qual progenitor será herdado cada gene.

Progenitor 1:	0	**0**	0	**1**	1	1	**0**	**1**	0
Progenitor 2:	1	**0**	1	**1**	0	0	**0**	**1**	1
	*	**0**	*	**1**	*	*	**0**	**1**	*
Descendente 1:	0	**0**	1	**1**	1	0	**0**	**1**	1
Descendente 2:	1	**0**	0	**1**	0	0	**0**	**1**	0

Figura 12 – *Half-uniform Crossover*

- *Order Crossover (OX1)*

O operador *Order Crossover* constrói descendentes escolhendo uma parte do cromossomo de um progenitor e preservando a ordem relativa dos elementos do outro progenitor. Por exemplo, supondo dois pontos de cortes nos progenitores da Figura 13, um entre a segunda e terceira posição e outro entre a sétima e oitava posição. Então os seguimentos entre os pontos de corte são copiados para dentro dos descendentes, como destacado em negrito. Começando do segundo ponto de corte de um progenitor, o resto dos genes são copiados na medida em que aparecem no outro progenitor, também começando do segundo ponto de corte e omitindo os genes que já estão presentes. Quando é alcançado o fim do cromossomo, continua-se da sua primeira posição.

No exemplo, na oitava posição do progenitor 1, existe a possibilidade dos valores 8, 9, 1 e 2 ocuparem esta posição. Na posição 8 do progenitor 1, possue valor 8, no progenitor 2 o valor 8 já ocupa uma dentro da região de corte, logo o próximo a ser escolhido é o valor 9, que assim ocupa a posição 8 do descendente 1. Seguindo essa lógica, as outras posições são preenchidas.

Progenitor 1:	1	2	**3**	**4**	**5**	**6**	**7**	8	9
Progenitor 2:	2	4	**6**	**8**	**7**	**5**	**3**	1	9
Pré-Descendente 1:	*	*	**3**	**4**	**5**	**6**	**7**	*	*
Pré-Descendente 2:	*	*	**6**	**8**	**7**	**5**	**3**	*	*
Descendente 1:	8	1	**3**	**4**	**5**	**6**	**7**	9	2
Descendente 2:	4	9	**6**	**8**	**7**	**5**	**3**	1	2

Figura 13– *Order Crossover*

- *OrderBased Crossover (OX2)*

O operador *Order Based Crossover* seleciona várias posições de um progenitor. A ordem dos valores nessas posições selecionadas é imposta ao outro progenitor. Considere a Figura 14, suponha que no segundo progenitor a segunda, terceira e sexta posições são selecionadas (em negrito). Os valores nessas posições são 4, 6 e 5 respectivamente. No primeiro progenitor, esses valores estão presentes na quarta, quinta e sexta posições (em negrito). O primeiro descendente será igual ao progenitor 1 exceto na quarta, quinta e sexta posições. Nos espaços com asteriscos, criado a partir do progenitor 1, acrescenta-se os valores na mesma ordem em que aparecem no segundo progenitor, como pode ser visto no descendente 1. O descendente 2 é obtido invertendo os papeis no processo descrito acima.

Progenitor 1:	1	2	3	**4**	**5**	**6**	7	8	9
Progenitor 2:	2	**4**	**6**	8	7	**5**	3	1	9
	1	2	3	*	*	*	7	8	9
Descendente 1:	1	2	3	4	6	5	7	8	9
Descendente 2:	2	4	3	8	7	5	6	1	9

Figura 14 – *Order Based Crossover*

- *Cycle Crossover (CX)*

No operador *Cycle Crossover* são criados descendentes que possuem posições ocupadas por valores correspondentes de cada um dos progenitores. Escolhe-se a primeira posição de um dos progenitores para ser o primeiro valor do primeiro descendente. A partir dos progenitores do exemplo da Figura 15, supondo que seja feita a escolha aleatória do valor 1 do primeiro progenitor como primeiro valor da primeira posição do futuro descendente, assim como aparece no pré-descendente da Figura 15. Se fosse escolhido a oitava posição do progenitor 2 para ser o oitavo elemento do descendente 1, resultaria em um cromossomo inválido onde um valor se repete. Para evitar essa repetição, a oitava posição é ocupada pelo valor correspondente do progenitor 1, escolhendo-se o valor 8. Escolhe-se o terceiro elemento de maneira análoga, resultando com o valor 4 na quarta posição do descendente 1, formando um ciclo 1 – 4, 4 – 8, 8 – 1.

Essa primeira parte do processo de formação de um descendente fecha um ciclo. Um segundo ciclo do operador se inicia escolhendo o restante dos elementos. O restante dos elementos serão as posições correspondentes do descendente 1 com o progenitor 2. Como mostra a Figura 15, a segunda, terceira, quinta, sexta, sétima e nona posições serão ocupadas pelos valores do progenitor 2. De maneira análoga, o descendente 2 é criado.

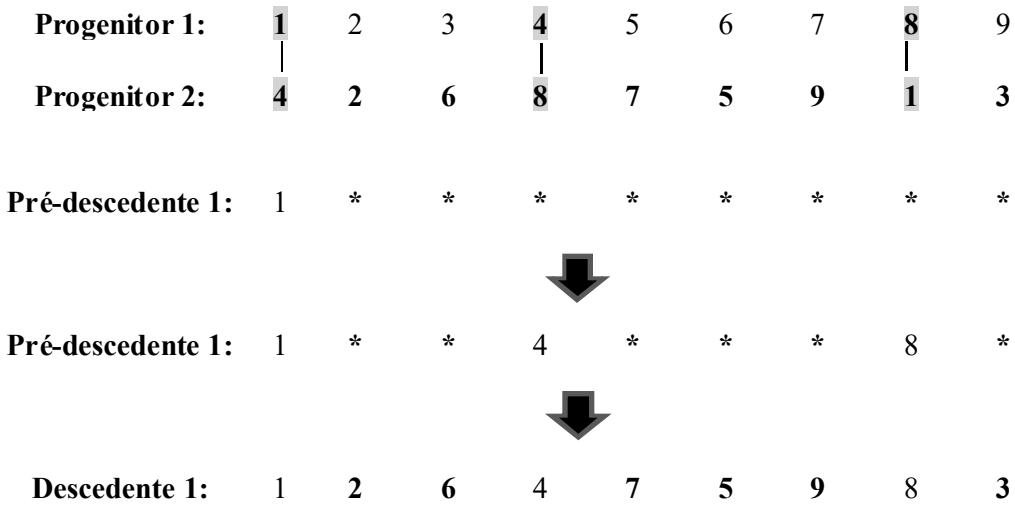

Figura 15 – *Cycle Crossover*

- *Partially Mapped Crossover (PMX)*

O operador *Partially Mapped Crossover* transmite as informações de ordenação e valores das cadeias do progenitor para os descendentes. Uma parte do cromossomo do progenitor é mapeada em uma parte do cromossomo do outro progenitor e as informações restantes são trocadas. Considerando como exemplo os progenitores 1 e 2 da Figura 16, o operador primeiramente seleciona aleatoriamente dois pontos de corte nos cromossomos dos progenitores. Supondo que o primeiro ponto de corte escolhido está entre a terceira e quarta posições, e o segundo entre a sexta e sétima posições. As subsequências entre os pontos de corte são chamadas de seções de mapeamento. No exemplo eles definem os mapeamentos 4 - 8, 5 - 7 e 6 - 5. Definida as sequências de mapeamento, elas são trocadas entre os progenitores, como demonstrado nos pré-descendentes da Figura 16, onde a seção de mapeamento do primeiro progenitor é copiada para o segundo descendente e a seção de mapeamento do segundo progenitor é copiada para o primeiro descendente.

A partir da troca da seção de mapeamento, o descendente 1 é preenchido com cópia dos elementos do progenitor 1. Caso um valor já esteja presente no progenitor, ele é substituído por outro, de acordo com o mapeamento. No exemplo, os valores das três primeiras posições do descendente 1 são copiados do progenitor 1, entretanto a sétima posição possui um valor já presente no descendente 1. Usando os mapeamentos definidos,

encontra-se o valor 7 mapeando 5, porém o valor 5 também já está pressente, dessa forma recorre-se ao mapeamento do valor 5, que é o valor 6. Na oitava posição do primeiro descendente utiliza-se o mapeamento do valor 8, que é o valor 4. O segundo descendente se forma de maneira análoga ao primeiro.

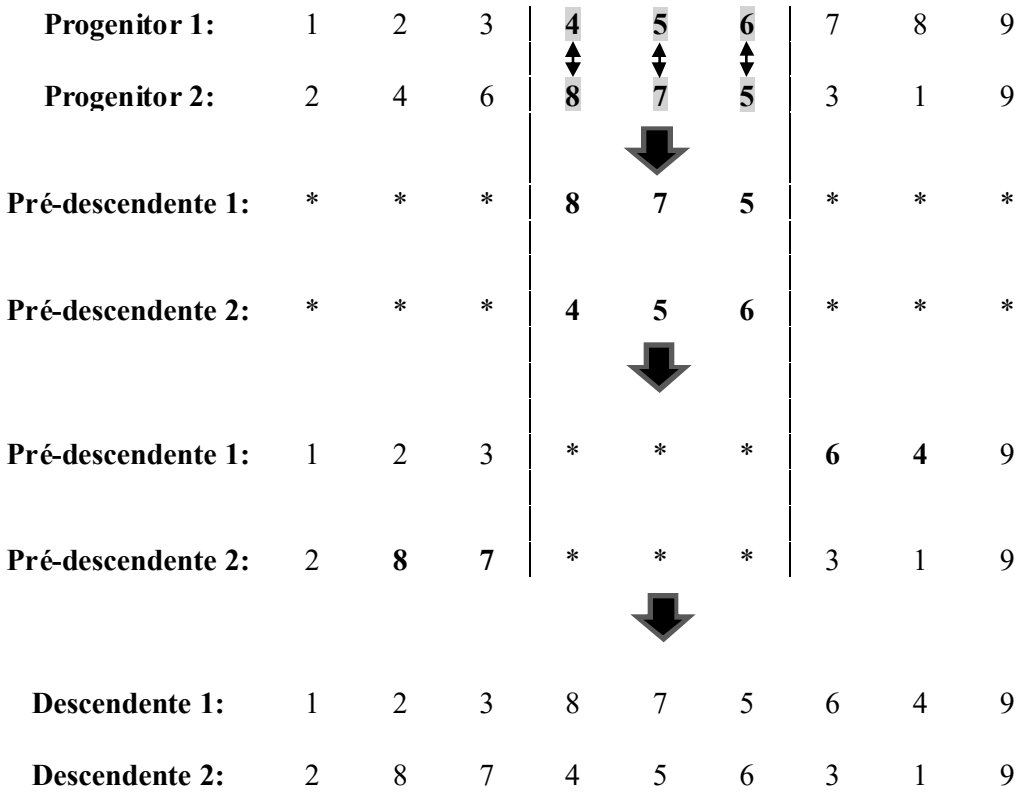

Figura 16 - *Partially Mapped Crossover*

- *Modified Partially-Mapped Crossover (MPMX)*

O operador *Modified Partially-Mapped Crossover* inicialmente divide os cromossomos dos progenitores e os descendentes em três seções aleatoriamente: esquerda, meio e direita. Essas seções são criadas aleatoriamente por meio da seleção de dois pontos de cruzamento aleatórios. A Figura 17 mostra um exemplo já com estas seções definidas. Na seção do meio, o descendente recebe os valores de um dos progenitores, no exemplo o descendente recebe os valores do progenitor 1.

A próxima etapa é a inserção de elementos nas seções esquerda e direita da descendência. Isso é feito usando o progenitor 2 como doador. Posições correspondentes no progenitor 2 doam elementos para o descendente, desde que eles ainda não tenham sido doados pelo progenitor 1. A etapa final é completar o descendente usando uma permutação aleatória dos elementos que ainda não foram alocados no descendente nas etapas anteriores. O descendente 2 é produzido de maneira análoga ao primeiro.

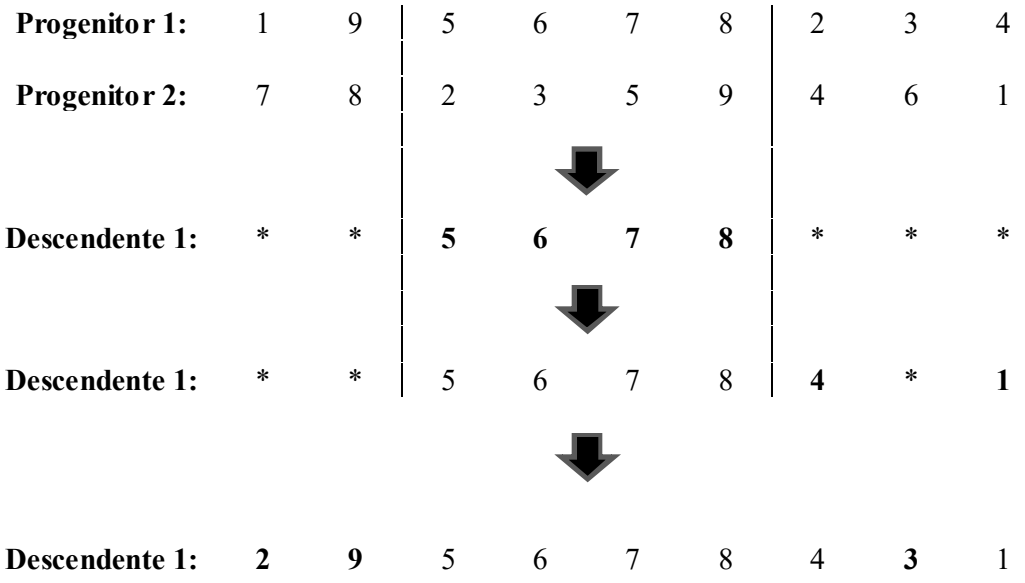

Figura 17 – *Modified Partially-Mapped Crossover*

- *Maximal Preservation Crossover (MPX)*

O operador *Maximal Preservation Crossover* opera inicialmente selecionando uma subsequência aleatória do primeiro progenitor. Esta subsequência possui um comprimento menor ou igual ao tamanho de posições do cromossomo dividido por 2. Um tamanho mínimo também é definido para essa subsequência uma vez que quando muito curtas são ineficazes e quando muito grandes não permitem uma variação significativa. A seleção do tamanho apropriado fornece um meio adequado para os progenitores transmitirem informações significativas para os descendentes. No exemplo da Figura 18, a subsequência selecionada do progenitor 1 é estabelecida da primeira posição até a quarta posição, sendo herdados pelo descendente 1 os valores correspondentes nessas posições.

O segundo estágio do operador é remover os elementos atualmente existentes no descendente do progenitor 2 a partir da subsequência selecionada no progenitor 1. No exemplo, os valores 1, 9 5 e 6 são removidos do progenitor 2. Em seguida, os valores restantes no progenitor 2 são inseridos no descendente 1, sendo a subsequência do primeiro progenitor colocada no início do descendente 1 e os valores livres remanescentes da descendência sendo preenchidos pelas cadeias restantes do progenitor 2.

Progenitor 1:	**1**	**9**	**5**	**6**	7	8	2	3	4
Progenitor 2:	7	8	2	3	5	9	4	6	1
Descendente 1:	**1**	**9**	**5**	**6**	*	*	*	*	*
Posições restantes do progenitor 2:	7	8	2	3	*	*	4	*	*
Descendente 1:	1	9	5	6	7	8	2	3	4

Figura 18 – *Maximal Preservation Crossover*

- *Position Crossover (PX)*

O operador *Position Crossover* está intimamente relacionado às técnicas de crossover OX1 e OX2. Na Figura 19 é apresentado um exemplo. O *Position Crossover* opera selecionando posições aleatórias ao longo das cadeias dos progenitores, no exemplo, as posições 1, 3, 4 e 8 são selecionadas. Os valores das posições selecionadas são então herdados pelos descendentes na ordem e nas posições em que ocorrem nos progenitores. O descendente 2 herdando do progenitor 1 e descendente 1 herdando do progenitor 2, gerando os pré-descendentes 1 e 2. Os elementos restantes necessários para completar os descendentes são doados pelos progenitores de origem, ou seja, descendente 1 herda do progenitor 1 e descendente 2 herda do progenitor 2, na ordem em que aparecem. Caso ocorra uma repetição é utilizado o mapeamento de acordo com as

posições selecionadas na parte inicial. No exemplo, a posição 6 do descendente 1 não pode herdar o valor 6 para evitar repetição, logo utiliza-se o mapeamento 3 – 6, utilizando o valor 3 na posição 6. O mesmo ocorre na posição 9 do descendente 2.

Progenitor 1:	1	2	3	4	5	6	7	8	9
Progenitor 2:	4	2	6	8	7	5	9	1	3
Pré-descedente 1:	**4**	*	**6**	**8**	*	*	*	**1**	*
Pré-descedente 2:	**1**	*	**3**	**4**	*	*	*	**8**	*
Descendente 1:	4	2	6	8	5	**3**	7	1	9
Descendente 2:	1	2	3	4	7	5	9	8	**6**

Figura 19 – *Position Crossover*

- *Voting Recombination Crossover (VR)*

O operador *Voting Recombination Crossover* não se inspira na biologia. Neste caso, pode ocorrer uma recombinação com diversos progenitores, sendo 2 o valor mínimo de progenitores. Na utilização do operador, é definido um número ***i*** natural menor ou igual ao número de progenitores, que se torna um parâmetro. Esse parâmetro identifica um mínimo de valores iguais para uma posição entre os progenitores e esses valores são copiados para o descendente.

A Figura 20 apresenta um exemplo deste operador, onde são utilizados 4 progenitores. No exemplo é definido ***i*** = 3. As posições 2, 3, 7 e 8 dos progenitores apresentam as repetições para o valor de ***i*** = 3, e o valores dessas posições são copiadas para o descendente. O restante das posições é definido em sorteio aleatório, evitando a repetição de valores que já estejam presentes no descendente.

Progenitor 1:	1	2	3	4	5	6	7	8	9
Progenitor 2:	4	2	6	8	9	3	7	1	5
Progenitor 3:	8	2	6	7	5	4	9	1	3
Progenitor 4:	4	2	6	8	9	5	7	1	3
Pré-descedente:	*	2	6	*	*	*	**7**	1	*
Descendente:	**4**	2	6	**8**	**5**	**9**	7	1	**3**

Figura 20 – *Voting Recombination Crossover*

3.2.1.2 Operadores de Mutação

A mutação refere-se ao processo de gerar novos indivíduos partindo de um único cromossomo, introduzindo uma alteração aleatória no indivíduo. O operador de mutação tem a função de impedir que a busca fique presa em mínimos locais e mantendo a diversidade por meio de mudanças aleatórias no cromossomo, bem como auxiliar na exploração do espaço de busca (HOLLAND, 1992). A mutação deve ser utilizada com cautela pois pode causar distorção no indivíduo, podendo causar perda de informações relevantes. Por isso, utiliza-se comumente baixas probabilidades de mutação (OLIVEIRA, 2017). E assim como o operador de recombinação, o operador de mutação deve ser escolhido de forma a não violar as restrições do problema e da representação utilizada.

A mutação foi inicialmente proposta para representação com cadeias de cromossomos binários. Cada posição de cromossomo possui uma probabilidade de sofrer mutação, ou seja, uma inversão para o caso da representação binária. A Figura 21 apresenta um exemplo onde as posições 1 e 5 foram sorteadas de modo que sofressem mutação.

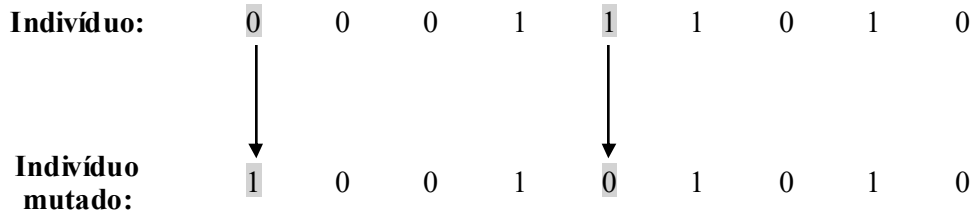

Figura 21 – Exemplo clássico de mutação

Os operadores de mutação apresentados nesta revisão da literatura envolvem a representação de permutações, se aproximando da representação cromossômica que se deseja utilizar neste trabalho, servindo de inspiração no desenvolvimento e adaptação de novos operadores. A Tabela 8 apresenta um resumo dos operadores de mutação que serão abordados nesta revisão.

Tabela 8 – Operadores de mutação considerados no estudo e adotados na literatura

Nome do Operador	Nomenclatura	Referência
Exchange Mutation	EM	(BANZHAF, 1990)
DisplacementMutation	DM	(MICHALEWICZ, 1992)
InsertionMutation	ISM	(FOGEL, 1988)
InversionMutation	IVM	(FOGEL, 1990)
SimpleInversionMutation	SIM	(HOLLAND, 1975)
ScrambleMutation	SM	(SYSWERDA, 1991)

- *Exchange Mutation (EM)*

O operador *Exchange Mutation* (mutação por troca) realiza a mutação de maneira simples, selecionando aleatoriamente duas posições no cromossomo e trocando seus valores. A Figura 22 apresenta um exemplo onde as posições 2 e 4 são selecionadas e a mutação ocorre com a troca dos seus valores.

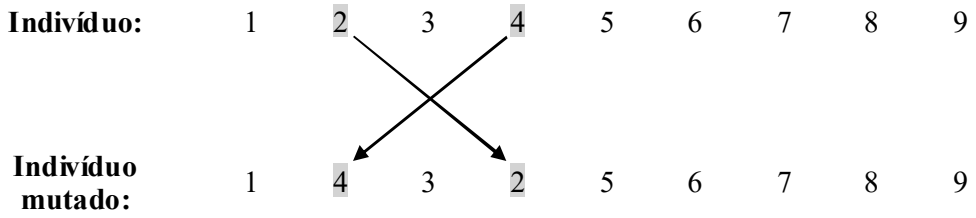

Figura 22 – Exchange Mutation

- *DisplacementMutation (DM)*

O operador de mutação *Displacement Mutation* (mutação por deslocamento) inicialmente seleciona aleatoriamente uma subsequência na cadeia do cromossomo. Esta subsequência é removida do cromossomo e inserida aleatoriamente em alguma outra posição do cromossomo. A Figura 23 apresenta um exemplo onde os valores das posições 4, 5 e 6 são selecionados e reinseridos entre as posições 8 e 9.

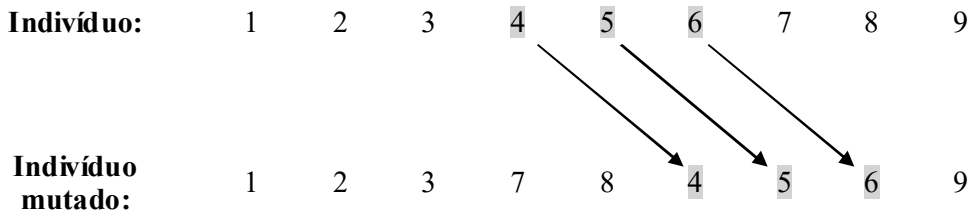

Figura 23 – *Displacement Mutation*

- *Insertion Mutation (ISM)*

O operador *Insertion Mutation* (mutação por inserção) escolhe aleatoriamente uma posição no cromossomo, remove o valor dessa posição e o insere aleatoriamente em algum lugar aleatoriamente escolhido no cromossomo. A Figura 24 apresenta um exemplo onde o valor da posição 2 é escolhido aleatoriamente e inserido entre as posições 4 e 5, também aleatoriamente.

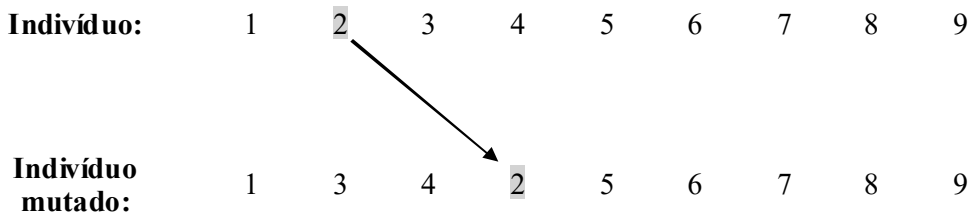

Figura 24 – *Insertion Mutation*

- *Inversion Mutation (IVM)*

O operador *Inversion Mutation* (mutação por inversão) funciona de maneira similar ao operador *Displacement Mutation* (mutação por deslocamento), onde também seleciona aleatoriamente uma subsequência do cromossomo e o reinsere aleatoriamente entra posição do cromossomo. Porém no caso do operador IVM a subsequência é invertida quando reinserida. A Figura 25 apresenta um exemplo onde os valores das posições 4, 5 e 6 são selecionados e reinseridos entre as posições 8 e 9. Quando reinseridos, a ordem da subsequência é invertida.

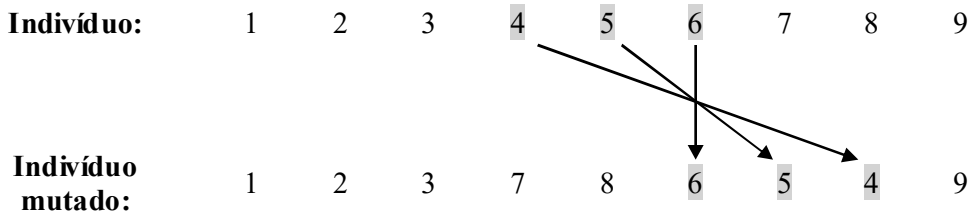

Figura 25 – *Inversion Mutation*

- *Simple Inversion Mutation (SIM)*

O operador *Simple Inversion Mutation* (mutação por inversão simples) seleciona aleatoriamente dois pontos de corte no cromossomo, separando uma subsequência. A subsequência selecionada é revertida entre os dois pontos selecionados. A Figura 26 mostra um exemplo onde o primeiro ponto de corte está entre as posições 1 e 2, e o segundo ponto de corte está entre as posições 5 e 6. A subsequência entre esses pontos é

invertida e reinserida no mesmo local de origem, ou seja, entre o primeiro e segundo ponto de corte.

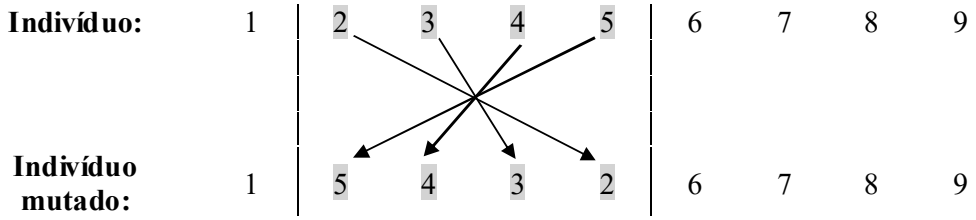

Figura 26 – *Simple Inversion Mutation*

- *Scramble Mutation (SM)*

O operador *Scramble Mutation* seleciona aleatoriamente uma subsequência no cromossomo e mistura aleatoriamente os valores contidos nessas posições. Após misturados, são reinseridos na posição de origem da subsequência. A Figura 27 mostra um exemplo onde uma subsequência da segunda até a quinta posição são selecionados, embaralhados e reinseridos no cromossomo.

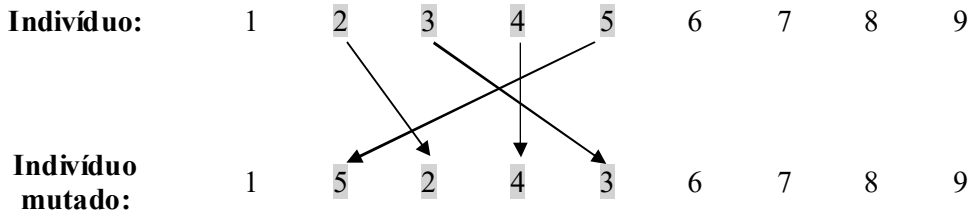

Figura 27 – *Scramble Mutation*

3.2.2 Evolução Diferencial e uma Alternativa para Representação Combinatória

O algoritmo de ED é uma heurística de busca estocástica de base populacional cuja finalidade é encontrar o indivíduo mais apto. A ideia nesse método é manter uma população de soluções candidatas e sintetizar novas combinando as soluções existentes usando operadores de mutação diferencial, recombinação e seleção trabalhando em sequência durante gerações mantendo o princípio elitista, isto é, mantém as soluções candidatas que alcançam o melhor valor de aptidão. Ele foi proposto pela primeira vez em otimização com variáveis contínuas em (PRICE; STORN, 1997), (PRICE, 1999) e

(PRICE et al, 2005) e hoje é um importante e poderoso mecanismo para otimização de objetivo único (PRICE et al, 2005), (FEOKTISTOV, 2006), (MEZURA-MONTES et al, 2006), (CHAKRABORTY, 2008), possuindo também aplicações em problemas multiobjetivo (XUE et al, 2003), (GONG; CAI, 2008), (BATISTA et al, 2009). O algoritmo DE foi inicialmente proposto para otimização contínua, porém ele foi aplicado com sucesso em problemas de otimização combinatória em (ONWUBOLU, 2006), (PAN et al, 2009), (QIAN et al, 2008).

Alguns problemas de otimização envolvem variáveis de otimização discretas que são números inteiros com significado numérico. Por outro lado, alguns problemas de otimização combinatória, como o TSP, envolvem variáveis de otimização de inteiros que são simbólicas, no sentido de que representam rótulos arbitrários, não representando qualquer quantidade numérica.

No cenário de otimização contínua, o DE pode ser aplicado com sucesso usando o truncamento, arredondando os valores reais para o próximo inteiro não-negativo, uma vez que as operações internas do algoritmo mantêm suas características numéricas. No entanto, em problemas com variáveis inteiras simbólicas, a operação principal de DE, o operador de mutação diferencial, não é mais aplicável, porque se baseia em operações aritméticas em variáveis que possuem natureza simbólica. O mecanismo de mutação diferencial emprega vetores diferenciais construídos com pares de vetores escolhidos aleatoriamente entre as soluções candidatas, a fim de produzir novas soluções. Em problemas contínuos de otimização, à medida que a população evolui, as direções de pesquisa e os tamanhos dos passos na mutação diferencial mudam ao longo do tempo de acordo com a distribuição da população no domínio da busca. O ED usa uma combinação de operadores clássicos de crossover e seleção com a mutação diferencial para evoluir uma população inicial aleatória para uma solução final de alta qualidade. Ao aplicar a mutação diferencial a problemas com variáveis simbólicas, os vetores diferenciais não apresentam uma correspondência direta e clara com as direções no espaço de busca. A subtração direta de soluções codificadas como vetores inteiros (vetores de permutação) não gera soluções viáveis e não representa nenhuma direção significativa, uma vez que a diferença não depende de propriedades numéricas, mas de rotulagem arbitrária (PRADO et al, 2014).

Uma nova abordagem geral para otimização combinatória é proposta em (PRADO et al, 2014) usando o algoritmo de ED e serve de inspiração para o trabalho aqui proposto uma vez que algoritmo participativo abordado utiliza princípios da ED. A abordagem

proposta visa preservar seu mecanismo de busca interessante para domínios discretos, definindo a diferença entre duas soluções candidatas como uma Lista Diferencial de Movimentos (LDM) no espaço de busca. Assim, um operador de mutação diferencial mais significativo e geral para o contexto de problemas de otimização combinatória pode ser produzido. Desta forma, conforme exigido pelo operador de mutação diferencial, os movimentos na lista diferencial podem ser então aplicados a outras soluções candidatas na população. À medida que a população evolui, o número e a diversidade de movimentos na lista diminuem, portanto, a mutação diferencial se autoadapta de acordo com a distribuição da população no domínio de busca, preservando sua capacidade em domínios discretos de adaptar as direções e tamanhos de passos na geração de novas soluções.

No sistema ED, novos indivíduos são gerados usando a mutação diferencial. A mutação diferencial segue de acordo com a equação (27) sendo baseada na diferença entre dois indivíduos escolhidos aleatoriamente da população atual. Esse vetor diferencial é multiplicado por uma constante e adicionado a um terceiro indivíduo, chamado vetor base.

$$v_{g,i} = x_{g,r_1} + F\left(x_{g,r_2} - x_{g,r_3}\right) \tag{27}$$

As definições da LDM definidas em (PRADO et al, 2014) necessárias para o algoritmo combinatório proposto em ED, estabelece operadores de soma, subtração e multiplicação na mutação diferencial. Elas são dadas abaixo:

Definição 1: Uma lista diferencial de movimentos $M_{j\to i}$ é uma lista contendo uma sequência de movimentos válidos m_k de tal forma que a aplicação desses movimentos para uma solução $S_j \in S$ leva à solução $S_i \in S$. Desta forma, a "diferença" entre duas soluções candidatas é definida como sendo esta lista de movimentos:

$$M_{j\to i} \doteq s_i \ominus s_j \,, \tag{28}$$

onde $\ominus$ é um operador binário negativo especial que retorna uma lista de movimentos $M_{j\to i}$ que representa um caminho de s_j em direção a s_i. Esta lista, em algum sentido, captura as diferenças entre essas duas soluções. A aplicação de uma lista de movimentos para uma determinada solução é definida da seguinte forma:

$$M_{j\to i} = s_i \oplus M_{a\to b}, \tag{29}$$

onde o operador binário $\oplus$ recebe uma solução válida e uma lista de movimentos, retornando outra solução. Observe que com essas definições, a seguinte relação é válida:

$$s_i = s_j \oplus M_{j\to i} = s_j \oplus (S_i \oplus S_j) \tag{30}$$

A multiplicação da lista diferencial de movimentos por uma constante também precisa ser definida e são apresentadas três alternativas distintas:

Definição 2: A multiplicação da lista de movimentos por uma constante $F \in [0,1]$, denotada como $F \otimes M_{j\to i}$, retorna uma lista $M_{j\to i}$com os primeirosmovimentos$\left|Fx\left|M_{j\to i}\right|\right|$ de $M_{j\to i}$, onde $\left|M_{j\to i}\right|$ é o tamanho da lista.

Definição 3: A multiplicação da lista de movimentos $\left|M_{j\to i}\right|$por uma constante $F \in [0,1]$,, denotada como $F \otimes M_{j\to i}$, retorna uma lista $M'_{j\to i}$. Para construir $M'_{j\to i}$., A lista $M_{j\to i}$é digitalizada do primeiro ao último movimento na lista, selecionando cada movimento $M'_{j\to i}$com probabilidade$F$.

Definição 4: A multiplicação da lista de movimentos$M_{j\to i}$ por uma constante $F \in [0,1]$, denotada como $F \otimes M_{j\to i}$retorna uma lista $M'_{j\to i}$ com $\left|Fx\left|M_{j\to i}\right|\right|$movimentos escolhidos aleatoriamente de $M_{j\to i}$, em uma seqüência aleatória, onde $M'_{j\to i}$ é o tamanho da lista.

Assim, a multiplicação da lista de movimentos por uma constante pode ser denotada, usando o operador de multiplicação binário especial $\otimes$, por:

$$M'_{j\to i} = F \otimes M_{i\to j}, \tag{31}$$

Como:

$$v_{g,i} = x_{g,r_1} \oplus F \otimes \left(x_{g,r_2} \ominus x_{g,r_3}\right) \tag{32}$$

$$v_{g,i} = x_{g,r_1} \oplus F \otimes M_{r_3\to r_2} \tag{33}$$

$$v_{g,i} = x_{g,r_1} \oplus M'_{r_3 \to r_2} \tag{34}$$

Estas operações representam a generalização da equação de mutação diferencial (27) de domínios contínuos para problemas de otimização combinatória.

Em (PRADO et al, 2014) o método proposto foi aplicado em problemas combinatórios conhecidos como TSP e geraram resultados preliminares indicando que a metodologia proposta é capaz de encontrar melhores resultados com melhor convergência. A abordagem proposta ainda tem a vantagem de ser aplicável a problemas gerais de otimização discreta, não apenas baseados em permutação, formando uma estrutura geral e unificadora para usar a evolução diferencial neste contexto.

Uma possível aplicação do método proposto em (PRADO et al, 2014) seria a utilização das LDM aplicadas diretamente sobre a compatibilidade e os operadores de recombinação e mutação utilizados no modelo de Aprendizado Participativo desenvolvido em (LIU, 2016). Esses operadores e a compatibilidade foram baseados em uma operação similar à aquela encontrada na ED, de forma que sua aplicação se encaixasse de forma conveniente.

4 Modelo Proposto

Conforme visto no Capítulo 1, uma tendência de pesquisa que se torna promissora é a Aprendizagem Participativa (AP). No desenvolvimento realizado em (LIU, 2016) diversas instâncias de algoritmos de Busca Participativa (BP) dentro da área de AP foram elaboradas e em particular a instância PSAR se destaca. Ela apresenta um desempenho superior às diversas instâncias de sistemas de busca existentes do estado da arte, além de apresentar um número reduzido de parâmetros a serem ajustados em sua execução (tamanho da população e número de gerações).

No Capítulo 2 foi apresentada uma outra tendência de pesquisa que se estabelece como uma estrutura alternativa aos SFBR, as Árvores de Padrão Fuzzy (APF) (HUANG; GEDEON; NIKRAVESH, 2008). As APF apontam características interessantes em seu modelo, tais como: possuir um modelo de seleção de atributos que auxilia no tratamento da "maldição da dimensionalidade"; são atraentes do ponto de vista de interpretação do resultado obtido pois apresentam uma "descrição lógica" de cada classe; possuem uma estrutura hierárquica que permite identificar quais variáveis são mais importantes.

Conforme as tendências apresentadas, propõe-se a utilização de algoritmos de BP, em especial o PSAR, para a indução de APF. Os algoritmos de BP são algoritmos de busca globais capazes de explorar grandes espaços de forma eficiente. A exploração do espaço de busca nesse caso não segue a estratégia gulosa do *Beam Search,* utilizada nos algoritmos de indução de APF, portanto tem mais chances de obter soluções melhores. Além disso, a exploração do espaço não é feita de forma restrita como no caso da *Beam Search* (que é limitada pela largura do feixe), o que também aumenta as chances de obter soluções melhores. A utilização de algoritmos de BP também facilita a utilização de operadores parametrizados, pois são capazes de alterar os parâmetros desses operadores com o objetivo de melhorar o desempenho da APF.

No Capítulo 3, algumas alternativas na resolução de problemas combinatórios foram vistas. A representação que se deseja utilizar nas APF, como sendo uma lista de números inteiros com restrições, semelhante à utilizada na PGC (SANTOS, 2014), se encaixa dentro da classe de problemas combinatórios. Dessa forma, alternativas em algoritmos e heurísticas na resolução de problemas combinatórios se faz necessária. Nesse contexto, se propõe uma reinterpretação dos operadores participativos utilizados na estrutura de funcionamento do PSAR, uma vez que eles não se encaixam diretamente na representação das APF, pois funcionam de forma aritmética. Outra alteração proposta

é a modificação na compatibilidade entre indivíduos, que segue método semelhante à Distância de Edição (NAVARRO, 2001).

A seguir serão descritas as etapas necessárias para a síntese das APF e as escolhas de projeto realizadas. A Figura 28 apresenta um diagrama de blocos que mostra, de uma forma geral, o método de síntese dos modelos aqui propostos. Em um primeiro momento, cada banco de dados utilizado será dividido em grupos de treinamento, validação e teste. Após a divisão dos grupos, os dados são fuzzificados e usados na síntese das APF. Com o intuito de se evitar *overfitting* (ocorre quando o modelo aprende muito sobre os dados utilizados no treinamento e perde capacidade de generalização) utilizou-se o conjunto de validação com parada antecipada. Após essa fase, utiliza-se o conjunto de teste para obter uma medida de generalização das APF sintetizadas. As principais diferenças e pontos do modelo desenvolvido serão descritas ao longo do capítulo.

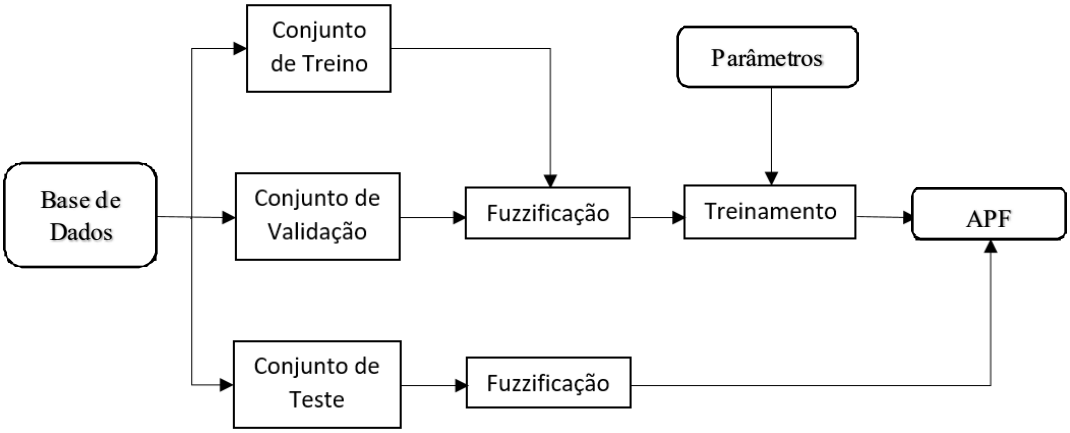

Figura 28 – Diagrama de blocos na síntese das APFs

4.1 Árvore e representação utilizada

A representação cromossômica que se pretende utilizar no desenvolvimento desse modelo segue as mesmas características do desenvolvido em (SANTOS, 2014) e (ALMEIDA, 2017), em que baseado na PGC (MILLER, 2011), um cromossomo é uma sequência linear de inteiros, representados por uma grade de nós computacionais de n-dimensões, podendo ser interpretados como programas escritos na forma de grafos. A representação do indivíduo é crucial para o desempenho do algoritmo, pois ela guia a escolha dos operadores genéticos e a função de avaliação. Não existe uma regra absoluta

determinística a ser seguida na construção de um operador de cruzamento (DRÉO, 2006). A ideia é que os parâmetros para construção de tal operador sejam guiados pelo domínio e restrições do problema em questão (SILVA, 2011).

Na Figura 29 é possível observar um exemplo de representação dos cromossomos (genótipos) e fenótipos (APF gerada pelo genótipo). A sequência linear de números inteiros é o cromossomo, onde cada número inteiro é chamado de gene, que pode ser rotulado de função, conexão ou saída. Cada nó é representado pela junção destes três tipos de genes, dos quais o gene de função utiliza os genes de conexão para gerar uma saída através de um conjunto de funções previamente definido. O primeiro gene se refere ao operador do nó ao qual pertence, e o segundo e terceiro genes do nó se refere às entradas do nó ao qual pertence.

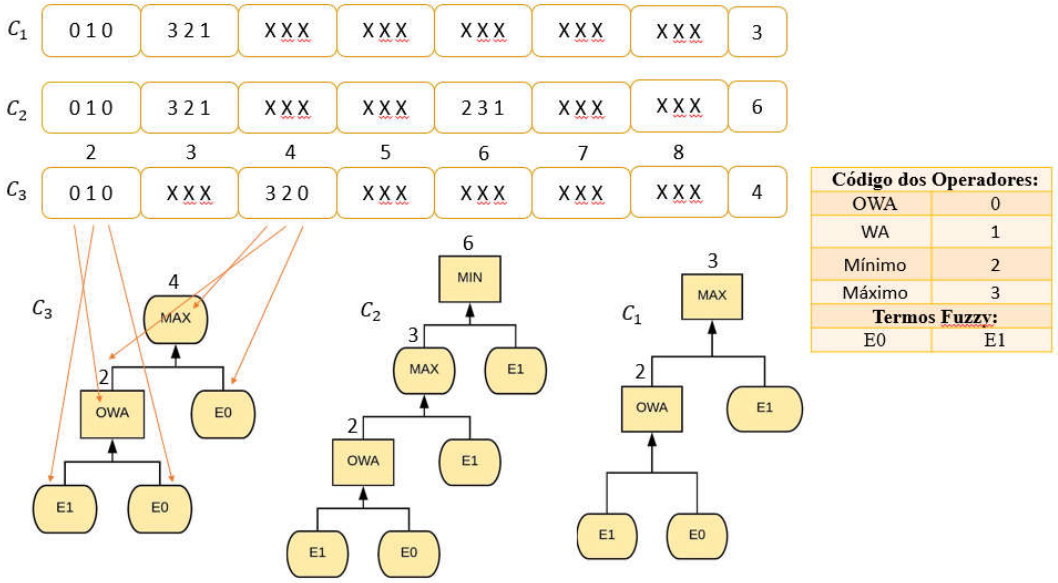

Figura 29 – Representação de Genótipos e Fenótipos

No exemplo da Figura 29 são apresentados 3 exemplos de genótipos (cromossomos) C_1, C_2 *e* C_3. Os conjuntos de 3 genes formam um nó da árvore onde, 2 genes representam as entradas desse nó e 1 gene o operador que esse nó utiliza. O último gene do cromossomo representa a saída da árvore. A marcação X no indivíduo representa um nó inativo, ou seja, que não se conecta a uma estrutura de nós que leve a saída da árvore. O cromossomo C_3 tem sua representação de fenótipo (APF) indicada com setas, formando sua árvore correspondente, assim como também é apresentada a representação

de fenótipo dos cromossomos C_1 *e* C_2. Na Figura 29 também é apresentada uma tabela com o código dos operadores *fuzzy* e os termos *fuzzy* associados às entradas do problema.

Alguns exemplos de funções podem ser encontrados na Tabela 9. A entrada de cada nó pode ser uma entrada do sistema ou a saída de um nó anterior, porém nunca do mesmo nó ou do nó à frente.

Tabela 9 – Exemplo de funções

Código do operador	Função que representa
0	OWA (*Ordered Weighted Average*)
1	WA (*Weighted Average*)
2	Mínimo
3	Máximo

Existem alguns parâmetros, herdados da PGC e referentes às APF, que devem ser definidos pelo usuário, tais como: *Levelback*, *Aridade*, número de nós e número de linhas. Assim como já explicado na Seção 2.5.3, esses parâmetros seguem as mesmas características representativas da PGC na síntese das APF. O *Levelback* é o máximo de nós anteriores que um gene de conexão pode se conectar. Nesse modelo foi fixado o número de linhas em 1, assim o tamanho do genótipo irá mudar somente com a variação do número de nós, sendo que cada nó possui 3 genes. O primeiro gene do nó, refere-se a um dos operadores. Sendo assim, a quantidade total de genes em um genótipo será igual ao número de nós vezes três, mais o gene de saída. A quantidade de entradas dos nós é chamada de *Aridade*, e é determinada de acordo com a função que necessita do maior número de entradas entre as funções do conjunto, um exemplo é apresentado na Figura 30 com *Aridade* 2 e 3 em um nó. Tanto o *Levelback* como a *Aridade*, são parâmetros a serem definidos pelo programador (MILLER; HARDING, 2009; MILLER; THOMSON, 2000; MILLER, 2011).

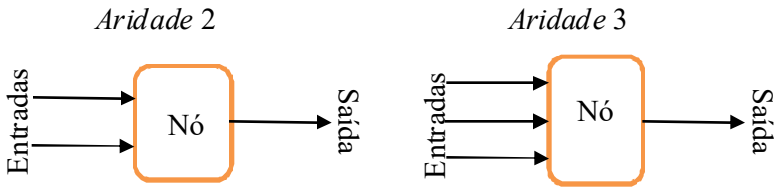

Figura 30 – Exemplo do funcionamento do parâmetro Aridade

O processo evolutivo ocorre nos genótipos, sendo necessária uma decodificação deste genótipo em fenótipos, cuja formatação está no domínio da solução do problema.

Quando o genótipo é decodificado, alguns nós podem estar desativados. Isso acontece, pois alguns nós não estão ligados à saída de dados, o que gera um efeito de neutralidade. Enquanto os genótipos tem um tamanho fixo, os fenótipos podem variar de tamanho, afetando a interpretabilidade do modelo.

4.2 Partições Fuzzy

Todos os bancos de dados utilizados na síntese das APF são fuzzificados para serem usados tanto na síntese (possíveis grupos de treinamento e validação) quanto nas entradas (grupo de teste) da APF. Nos modelos propostos nesse trabalho, os atributos de entrada são particionados em conjuntos *fuzzy*.

A Figura 31 exemplifica a partição de um atributo, cujo domínio é o intervalo [0,1], onde cada atributo teve seu domínio dividido em 5 termos linguísticos (termos *fuzzy*) denominados “Baixo”, ”Médio-Baixo”, “Médio”, “Médio-Alto”, “Alto”. O valor de cada atributo ativa a função de pertinência de cada termo *fuzzy* produzindo um valor de pertinência no intervalo [0,1]. Uma vez que a quantidade de termos linguísticos pode ser escolhida e influencia diretamente na quantidade de possíveis entradas na APF, o número de termos utilizados se torna um parâmetro a ser estabelecido. O método aqui proposto pretende realizar variações de cenários em relação à quantidade de termos linguísticos na síntese de APFs, com o objetivo de encontrar o que obtém melhor desempenho.

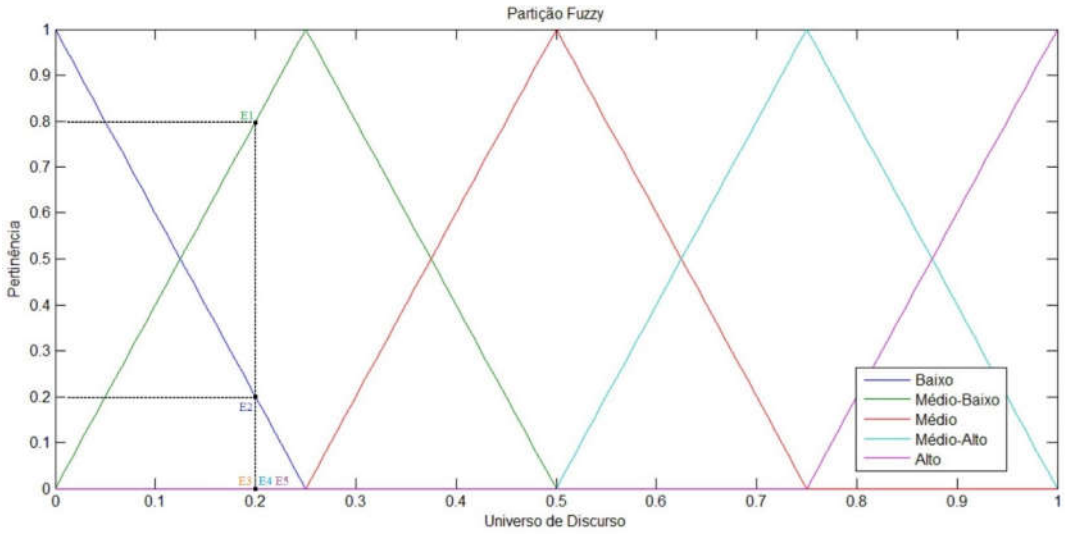

Figura 31 – Partições Fuzzy
Adaptado de (SANTOS, 2014)

Na Figura 31, observa-se que cada atributo presente no banco de dados terá um valor de pertinência associado a cada um dos termos *fuzzy*. Caracterizando o número de entradas possíveis na APF como sendo o número de atributos do banco de dados em uso, multiplicado pelo número de termos *fuzzy* em uso. O exemplo que é apresentado, tem no universo de discurso, de um atributo de entrada, um valor de 0.2, dessa forma, ele gera 5 possíveis entradas para uma APF: E1, E2, E3, E4 e E5. No exemplo somente os termos *fuzzy* "baixo" e "médio-baixo" geram valores não nulos de entrada.

É fundamental que o modelo proposto possa ser aplicado a novos casos além dos usados em sua fase de treinamento, sendo desejável que não se especialize somente na classificação dos casos conhecidos, mas possa também classificar novos casos de forma igualmente eficiente ou se possível com melhor desempenho. Ou seja, trata-se de classificar os novos casos corretamente dado que os antigos foram usados no treino e previamente classificados.

Em muitos casos, o conjunto de validação é obtido dividindo-se o conjunto de treinamento e utilizando uma parte dele para o treinamento propriamente dito e a outra parte na parada antecipada. Nas situações em que a base de dados for pequena pode-se adotar uma outra estratégia: gerar este conjunto de validação artificialmente (DUIN, 1996).

Utilizou-se o conjunto de treinamento para estimar a função de densidade de probabilidade que gerou os dados do conjunto. A partir desta função de densidade de probabilidade, dados artificiais são gerados com características similares ao conjunto de treinamento. A estimativa de densidade de probabilidade pode ser feita através de Janelas de Parzen ou por vizinhos mais próximos (DUDA; HART; STORK, 2012).

Esses dados sintéticos gerados para o conjunto de validação devem atender a dois requisitos: primeiro, deve se assemelhar um pouco aos dados originais estatisticamente, para garantir o realismo e manter os problemas envolvidos. Em segundo lugar, também deve se assemelhar estrutural e formalmente aos dados originais, de modo que qualquer software escrito sobre ele possa ser reutilizado (PATKI, et al. 2016).

4.3 Operadores das APFs

No modelo de APF que se deseja gerar, diversos operadores podem ser utilizados na sua construção assim como já foi visto, tais como operadores *fuzzy T-Norm* e *T-conorm*. Em (SANTOS, 2014) foram realizados experimentos, utilizando base de dados artificiais, que demonstram que a utilização de operadores selecionados se torna mais eficiente em relação à utilização de todos os possíveis operadores da APF original. Utilizar um conjunto reduzido de operadores facilita a compreensão da expressão representada pela árvore obtida. Para isso foi utilizando como medidas de desempenho a acurácia e o AUC (área sob a curva ROC), constatado, através do teste estatístico não-paramétrico Friedman (DERRAC et al., 2011) e do teste de comparação múltipla Nemenyi para um grau de confiança de 95%, que não existe diferença estatisticamente significativa entre o conjunto completo e com o conjunto reduzido de operadores. Cabe ressaltar que para problemas com mais de duas classes, a AUC é a média das AUCs utilizando a estratégia 1 contra todos. Dessa forma, o modelo aqui proposto, utiliza os seguintes operadores:

$$Máximo = \max(a, b) \tag{35}$$

$$Mínimo = \min(a, b) \tag{36}$$

$$WA = x \times a + (1 - x) \times b \tag{37}$$

$$OWA = x \times \max(a, b) + (1, x) \times \min(a, b) \tag{38}$$

Os operadores WA e OWA são parametrizados, onde a e b são os valores de entradas dos nós que serão operados e x será um valor aleatório dentro do intervalo [0,1], que poderá ser ajustado pelo operador de mutação. Os operadores WA e OWA são utilizados para preencher o espaço entre a maior combinação conjuntiva (*t-norma*) e a menor combinação disjuntiva (*t-conorma*). Os operadores foram codificados em números inteiros de acordo com a Tabela 10.

Tabela 10 – Código no cromossomo dos operadores utilizados

Operador	Código do operador
WA	0
OWA	1
Mínimo	2
Máximo	3

No modelo aqui adotado, a *Aridade* fica limitada em 2, uma vez que os operadores selecionados se utilizam de 2 entradas em suas funções.

4.4 Aptidão

A avaliação da população é realizada pela função de aptidão, que deve indicar a "qualidade" de cada indivíduo (genótipo) na população. A escolha da função de aptidão é para a maioria das aplicações a etapa crítica do processo, já que ela deverá ser avaliada para cada cromossomo de cada população dentro do processo evolutivo e define o sucesso no processo de busca. A avaliação dos genótipos proposta é composta por duas parcelas como pode ser visto na Equação (39). Ela apresenta a forma mais simples de função de aptidão multiobjetivo, que é aquela composta por ponderação de objetivos. Esses tipos de algoritmos utilizam o conceito de dominância de Pareto (AIMIN, 2011), que permitem comparar duas soluções além de uma única perspectiva (ou objetivo). Aqui será adotada uma ponderação simples entre os objetivos acurácia e interpretabilidade da seguinte forma:

$$Aptidão = W_1 \times AP_1 + W_2 \times AP_2, \tag{39}$$

onde W_1 e W_2 são pesos a serem atribuídos, desde que $W_1 + W_2 = 1$.

A primeira parcela AP_1 refere-se à função de avaliação que trata da acurácia. Diversas funções podem ser utilizadas, como por exemplo, Área sob a Curva ROC (FAWCETT, 2006), F-Measure (FERRI; HERNÁNDEZ-ORALLO; MODROIU, 2009), RMSE (WITTEN; FRANK, 2005) ou a Entropia Cruzada (BAUGHMAN; LIU, 1995). A similaridade obtida a partir do RMSE (*Root Mean Squared Error*), que foi utilizada em algoritmos anteriores, é uma medida baseada em distância, definida pelas equações (40)

e (41). Em problemas de classificação a função de custo Entropia Cruzada, definida pela equação (42), obtém resultados positivos (BAUGHMAN; LIU, 1995).

$$Rmse = \sqrt{\frac{\sum_{i=1}^{N}(d_i - o_i)^2}{N}}, \tag{40}$$

$$Similaridade = 1 - Rmse, \tag{41}$$

$$Entropia\ Cruzada = \frac{1}{N}\sum_{N} d_i \times \ln o_i + (1 - d_i) \times \ln(1 - o_i), \tag{42}$$

O RMSE produz como resultado a raiz quadrada do erro quadrático médio, que é definido a partir da diferença entre valor obtido na saída da árvore (o_i) e o valor alvo (d_i), que deve ser "1" se os valores apresentados na entrada pertencem a um ponto da classe que a árvore deve representar e "0" caso contrário. O RMSE é subtraído de um, resultando na medida de desempenho chamada similaridade. A segunda parcela AP_2 da avaliação dos genótipos é descrita pela equação (43). Ela procura representar a interpretabilidade

$$AP_2 = \frac{Número\ Total\ de\ Genes - Genes\ Ativos}{Número\ Total\ de\ Genes}, \tag{43}$$

A equação (43) produz um valor maior, quanto menor for a quantidade de genes ativos em comparação a quantidade total de genes. Os genes ativos são aqueles para os quais a saída está conectada a algum outro gene. Esta segunda parcela fornece uma melhor avaliação para árvores menores. Quanto maior for uma árvore, mais complexa será a expressão "lógica" que representará a classe e mais difícil será a compreensão da expressão obtida.

A função de avaliação que foi utilizada gera 2 parâmetros de entrada para o algoritmo. Em (SANTOS, 2014) foi utilizada a mesma função para a avaliação das APF e um estudo de casos comprovou que a proporção $W_1 = 0{,}7$ e $W_2 = 0{,}3$ apresenta melhor desempenho além de limitar o tamanho da árvore. Nesse estudo foram verificadas as proporções $W_1 = 1$ e $W_2 = 0$; $W_1 = 0{,}7$ e $W_2 = 0{,}3$; $W_1 = 0{,}5$ e $W_2 = 0{,}5$; $W_1 = 0{,}3$ e $W_2 = 0{,}7$. Verificou-se que, conforme se aumenta a limitação do tamanho da árvore, o

desempenho diminui. A partir dos resultados obtidos em (SANTOS, 2014) serão utilizadas aqui as proporções de W_1 eW_2.

4.5 Algoritmos de busca propostos na síntese das APF

Nesta seção serão apresentados 4 modelos baseados na Busca Participativa para síntese das APF utilizando as características do algoritmo PSAR como inspiração. Os métodos propostos utilizam a representação das APF de acordo com a Seção 4.1, as partições *fuzzy* expostas na Seção 4.2, os operadores *fuzzy* demonstrados na Seção 4.3 e a função de aptidão da Seção 4.4.

Todos os modelos propostos buscam explorar alternativas de utilização da busca participativa na síntese das APF, procurando manter o desempenho do método participativo original. A seguir, na Tabela 11, é apresentado um resumo com as principais características de todos os modelos propostos neste trabalho, assim como a nomenclatura adotada para cada modelo.

Tabela 11 – Resumo dos Algoritmos de Busca Propostos

Modelo proposto	Característica principal
BP_APF_AP	Utiliza aproximações para enquadrar o método numérico com números racionais do PSAR à representação cromossômica adotada para as APF com números inteiros.
BP_APF	Reinterpreta os operadores do algoritmo PSAR, inspirados nas diversas possibilidades disponíveis na literatura, de forma que consigam se adequar à representação cromossômica adotada das APF.
BP_APF_AG	Utiliza operadores de recombinação e mutação do Algoritmo Genético clássico na estrutura evolutiva do PSAR, substituindo os operadores numéricos originais.
BP_APF_TS	Utiliza a reinterpretação do modelo BP_APF em conjunto com o método de Transferência Seletiva, abordado na Seção 1.3.

4.5.1 Busca Participativa com Aproximações (BP_APF_AP)

O modelo desenvolvido BP_APF_AP pretende utilizar o modelo numérico integral do PSAR na busca das APF. Para isso, toda a estrutura do PSAR será utilizada sem alterações nos operadores de seleção, recombinação e mutação.

O proposto é utilizar aproximações de forma que o valor do gene, que varia de "0" a "1" no PSAR, se torne o valor mais próximo do correspondente, para representação adotada, em cada posição do cromossomo na representação das APF. Para isso, em cada gene é usada uma normalização, enquadrando o valor da posição no cromossomo do PSAR, pertencente ao intervalo [0 1], no intervalo de valores possíveis na representação das APF. Após a normalização é utilizado um arredondamento para o valor inteiro mais próximo. O processo pode ser dividido em duas etapas, na primeira a conversão do valor que varia no intervalo [0 1] do PSAR para o intervalo mínimo e máximo da posição do gene. A segunda etapa é transformar o número racional positivo no menor número inteiro mais próximo.

O tratamento de bases de dados, em áreas como a de mineração de dados e aprendizado de máquinas, tem como propósito minimizar os problemas oriundos do uso de unidades e dispersões distintas entre as variáveis. No caso em questão, a normalização será realizada segundo a amplitude das variáveis de acordo com a equação (44).

A Figura 32 apresenta um exemplo onde um cromossomo com 6 nós, 1 saída e 19 possíveis entradas da árvore (entradas fuzzificadas oriundas de uma base de dados) é normalizado e logo após tem seus valores arredondados. Na Figura 32 os vetores $Máximo^i$ e $Mínimo^i$ mostram os valores máximos e mínimos que cada posição (gene) do cromossomo pode assumir na representação das APF. C^i_{PSAR} representa o cromossomo a ser normalizado utilizado no algoritmo PSAR. $C^i_{PSAR\_norm}$ representa o cromossomo já normalizado e C^i_{APF} representa o cromossomo com seus valores arredondados após a normalização.

$$C^i_{PSAR\_norm} = (C^i_{PSAR} \times (Máximo^i - Mínimo^i)) + Minimo^i \qquad (44)$$

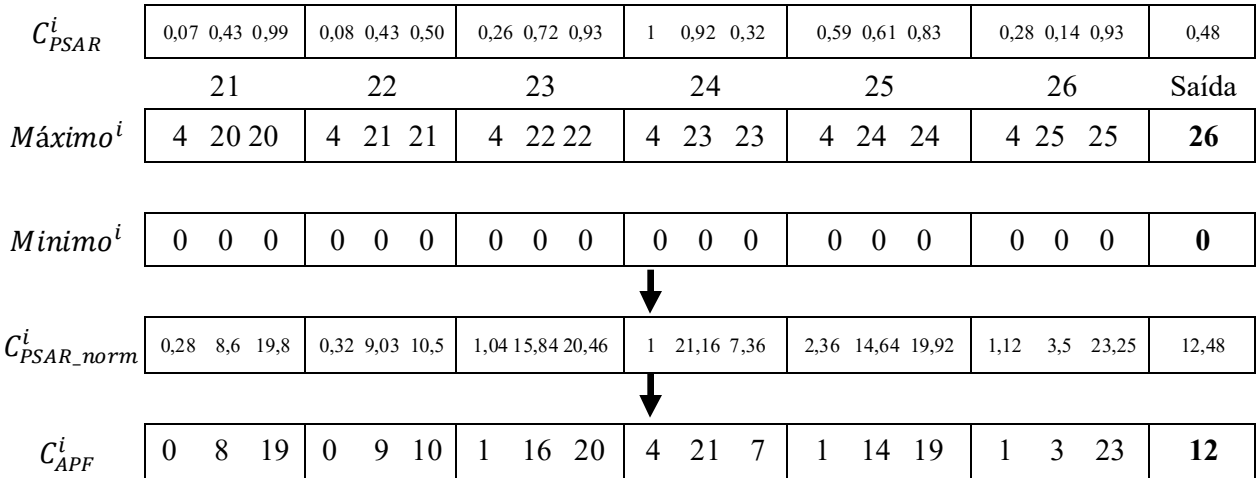

Figura 32 – Exemplo de normalização e arrendondamento no cromossomo do PSAR utilizado no modelo com aproximações

4.5.2 Busca Participativa com Operadores Reinterpretados (BP_APF)

O modelo desenvolvido BP_APF pretende reinterpretar os operadores de recombinação e mutação, e o cálculo de compatibilidade do PSAR, de forma que ele atue como um algoritmo de otimização combinatória atendendo aos requisitos da representação cromossômica adotada na Seção 4.1. Na representação adotada, uma lista de números inteiros com restrições características de cada gene representa um cromossomo.

A seguir, serão descritas por etapas como cada operador de compatibilidade, recombinação e mutação do algoritmo PSAR irá tratar o cromossomo. A partir do código conceitual do PSAR, apresentado logo abaixo, é possível afirmar que as mudanças ocorrem somente nas linhas 6 e 7, em relação ao cálculo da compatibilidade, e na linha 9, onde ocorrem a recombinação e mutação.

1: **procedimento** PSAR
2: Definir$t = 0$
3: Escolher$best^t$
4: **repetir**
5: Gerar S^t usandoμ_t
6: Gerar $S^{t\prime}$ a partir da compatibilidade entre os indivíduos de S^t
7: Gerar $p_{selected}$a partir da compatibilidade entre os indivíduos de S^t e$S^{t\prime}$ em relação ao $best^t$
8: Encontrar $best^t$ emS^t
9: Definir$best(S^t) = D(S^t, best^t) =$ $\begin{cases} p_r^t = (1 - \alpha{\rho_r}^t)s^t + (\alpha{\rho_r}^t)s^{t'} \\ p_m^t = best^t + {\rho_m}^t(p_{selected}^t - p_r^t) \\ best^t \end{cases}$
10: Definir$t = t + 1$
11: **até** critério de parada ser alcançado
12: **fim do procedimento**

4.5.2.1 Seleção e compatibilidade

O operador de seleção independe de como o indivíduo é representado dado que somente uma medida de compatibilidade entre cromossomos se faz necessária para que a seleção gere o grupo $p_{selected}$ e ocorra conforme descrito na seção 1.2.2. No método originalmente proposto do PSAR, extraído da equação (45), a valor da compatibilidade deve estar no intervalo [0,1]. Propõe-se uma medida de compatibilidade que respeite esse intervalo e que possa fornecer um grau de compatibilidade entre dois cromossomos com a representação adotada.

$$\rho_t = 1 - \frac{1}{n}\sum_{k=1}^{n} |z(t) - v(t)| \tag{45}$$

A medida de compatibilidade proposta avalia o número de genes iguais entre dois cromossomos e faz a média em relação ao número total de genes no cromossomo. Neste método proposto quanto maior o número de posições iguais entre os cromossomos, maior será a compatibilidade. Se todas as posições do cromossomo forem iguais, a compatibilidade será máxima, com valor 1 (um). No caso em que nenhuma posição entre os cromossomos é igual, a compatibilidade é 0 (zero).

Definido o método de cálculo da compatibilidade, a seleção prossegue como idealizada originalmente no PSAR, avaliando-se entre os grupos S_t (população gerada aleatoriamente na geração *t*) e S_t' (população gerada a partir da compatibilidade entre os indivíduos de S_t), que mais se aproxima do melhor indivíduo da geração atual $best_t$, gerando o grupo $p_{selected}$. A Figura 33 ilustra o processo de seleção que ocorre no PSAR, em que ρ^t representa a compatibilidade entre o indivíduo de S_t e o $best_t$, e $\rho^{t'}$ representa a compatibilidade entre $S_t{}'$ e o $best_t$. No caso de $\rho^t \geq \rho^{t'}$, o indivíduo pertencente ao grupo *S* irá compor o grupo $p_{selected}$, caso contrário o indivíduo pertencente ao grupo *S'* irá compor o grupo $p_{selected}$.

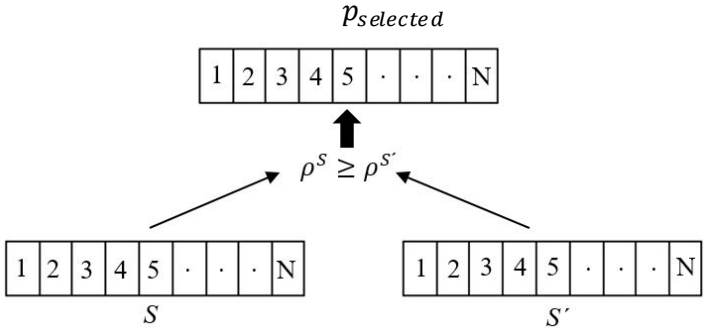

Figura 33 – Seleção no PSAR

A Figura 34 apresenta um exemplo da aplicação do cálculo de compatibilidade entre dois indivíduos utilizando a representação adotada. No exemplo, é gerado um vetor binário *S* onde as posições com valor "1" representam posições iguais entre os cromossomos C_1 e C_2 e "0" representam posições diferentes entre os cromossomos C_1 e C_2. A partir do vetor *S* gerado, é realizado o somatório dos valores e dividido pelo número total de posições existentes no cromossomo. O cálculo da compatibilidade nesse exemplo é apresentado na equação (46).

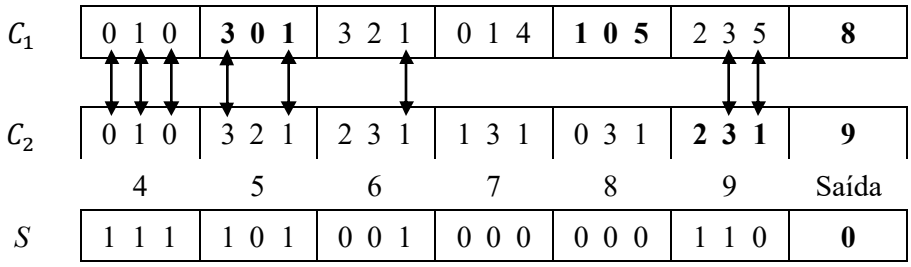

Figura 34 – Exemplo do cálculo de compatibilidade na síntese das APFs

$$\rho_{C_1,C_2} = \frac{1}{19}\sum_{k=1}^{19} S_k = \frac{8}{19} = 0{,}421 \quad (46)$$

4.5.2.2 Recombinação

A recombinação ocorre de acordo com a equação (47) apresentada a seguir, entre os indivíduos S_t e S_t' da geração *t*. O sistema de recombinação é modulado por um grau de compatibilidade ρ_t e um índice de excitação α. O grau de compatibilidade ρ_t expressa a compatibilidade entre os indivíduos que estão sendo recombinados na etapa *t* do algoritmo. O índice de excitação corresponde a uma taxa básica de aprendizado que possui um caráter de influência aleatório dado que seu valor é definido aleatoriamente no intervalo [0,1].

A recombinação gera o grupo p_r a partir dos indivíduos dos grupos S_t e S_t' sob influência direta da compatibilidade e um índice de excitação. Na recombinação a compatibilidade e o índice de excitação ditam o quanto de cada indivíduo dos grupos S_t e S_t' irão fazer parte do indivíduo recombinado, conforme as parcelas A e B da equação (48).

No processo numérico contínuo, originalmente proposto no PSAR, as parcelas A e B definem uma proporção de quanto cada indivíduo S_t eS_t' farão parte do novo indivíduo recombinado. Seguindo essa interpretação e utilizando a representação de números inteiros proposta, as parcelas A e B determinem o quanto dos indivíduos S_t eS_t' farão parte do indivíduo recombinado a partir de um ponto de corte. Dessa formaos cromossomos realizem uma recombinação com 1 ponto de corte, conforme modelo 1PX descrito na seção 3.2.1.1. A Figura 35 apresenta um exemplo do método combinatório proposto para $A = 0{,}37$ e $B = 0{,}63$. No exemplo, o valor de $A = 0{,}37$ define que 37% de S_t fará parte do indivíduo recombinado, enquanto 63% de S_t' fará parte do indivíduo recombinado. Para isso o ponto de corte se estabelece entre as posições 7 e 8 do cromossomo.

$$p_r = (1 - \alpha{\rho_t}^{1-a_t})s + (\alpha{\rho_t}^{1-a_t})s' \quad (47)$$

$$p_r = \underbrace{(1 - \alpha{\rho_t}^{1-a_t})}_{A} s + \underbrace{(\alpha{\rho_t}^{1-a_t})}_{B} s' \quad \in [0\ 1] \tag{48}$$

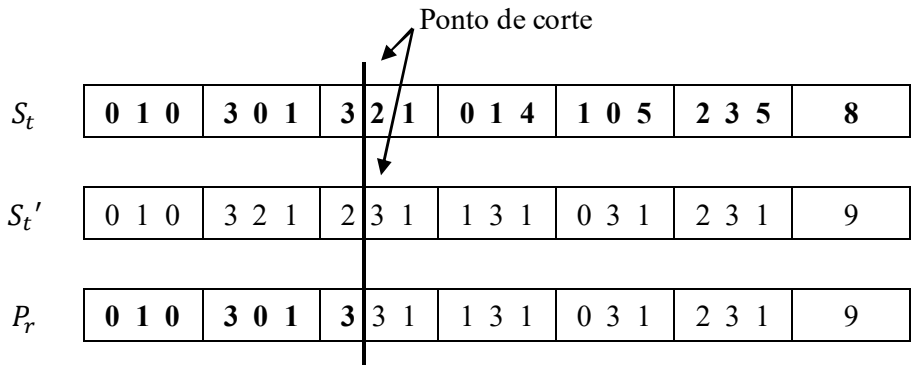

Figura 35 – Recombinação proposta

4.5.2.3 Mutação

A mutação no PSAR ocorre produzindo novos indivíduos adicionando diferenças ponderadas entre indivíduos dos grupos $p_{selected}$ e p_r ao melhor indivíduo $best_t$ da geração *t*. Esse método de mutação é semelhante ao que ocorre na Evolução Diferencial (PRICE et al., 1997). A mutação no PSAR produz os indivíduos do conjunto p_m de acordo com a equação (49) apresentada logo a seguir.

$$p_m = best_t + \underbrace{\rho_t{}^{1-a}}_{\in [0\ 1]}(\ p_{selected} - p_r) \tag{49}$$

O que se propõe é viabilizar a subtração, soma e multiplicação que ocorre na equação (49), de forma que essas operações se tornem viáveis para um vetor de números inteiros, respeitando as restrições da representação adotada. A proposta é dividir o

processo em duas etapas. Em um primeiro momento, realizar a subtração $(p_{selected} - p_r)$ entre $p_{selected}$ e p_r,e em um segundo, a multiplicação ${\rho_t}^{1-a}(\ p_{selected} - p_r)$entre o resultado da subtração e ${\rho_t}^{1-a}$, em conjunto com a soma em relação ao $best_t$.

Pode-se encarar a operação de subtração em *Pselected* e *Pr* como sendo um processo de encontrar a parte que falta para *Pselected* se tornar *Pr*. Dessa forma, a subtração para um vetor de inteiros será realizada como uma comparação entre cromossomos. As posições correspondentes entre dois indivíduos que são iguais serão mantidas. Posições correspondentes com valores diferentes serão sorteadas. O sorteio ocorre dentro dos limites que cada posição do cromossomo excluindo a possibilidade de os valores sorteados serem iguais ao contido nos indivíduos que compõe a subtração, ou seja, as restrições para cada posição serão mantidas respeitando o processo de construção das APFs dentro da representação cromossômica adotada. A Figura 36 ilustra o processo descrito através de um exemplo. No exemplo, as posições 1, 2, 3, 4, 6, 9,16 e 17 apresentam valores iguais que são mantidos no resultado da subtração. Nas outras posições são sorteados valores de forma que se respeitem as restrições da posição no cromossomo e que não ocorra um valor igual ao de $p_{selected}$ oup_r.

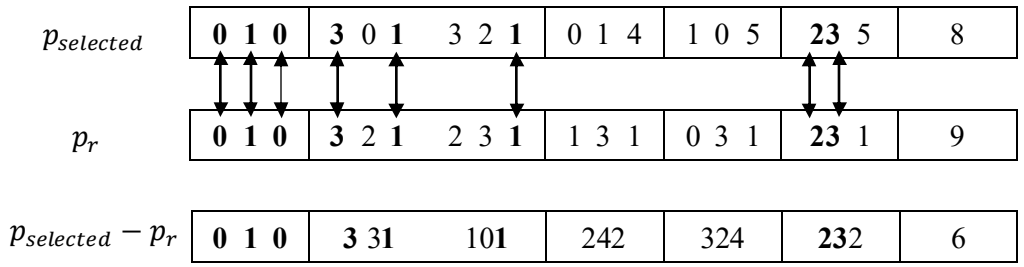

Figura 36 – Exemplo do processo de subtração proposto

A segunda etapa do processo proposto é a multiplicação em conjunto com a soma. Considera-se essa etapa de forma semelhante ao adotado na Recombinação. A parcela ρ_t^{1-a} pertence ao intervalo [0 1] de forma que ela representará o quanto do resultado da subtração entre *Pselected* e *Pr* irá fazer parte do melhor indivíduo $best_t$. A Figura 37 apresenta um exemplo do método proposto para um valor de $\rho_t^{1-a} = 0{,}26$. No exemplo, $\rho_t^{1-a} = 0{,}26$, de forma que 26% do resultado da subtração entre *Pselected* e *Pr* irá fazer

parte do indivíduo mutado p_m e 73% do $best_t$ irá fazer parte do indivíduo mutado p_m.Para isso um ponto de corte é definido entre as posições 5 e 6 dos cromossomos da mutação. E a mutação é realizada com troca de genes entre os cromossomos a partir do ponto de corte, assim como no processo de recombinação.

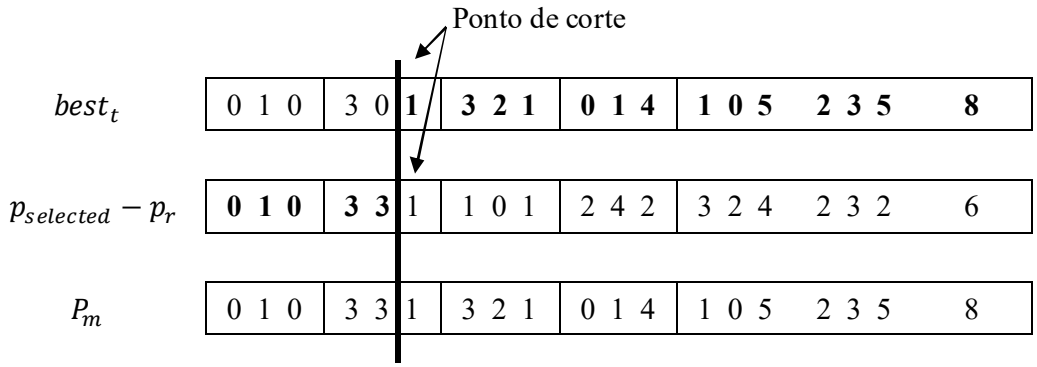

Figura 37 – Segunda etapa do processo proposto para mutação

4.5.3 Busca Participativa com Operadores Clássicos (BP_APF_AG)

O modelo desenvolvido BP_APF_AG pretende utilizar a estratégia evolucionária do PSAR com a representação adotada para as APF, alterando os operadores de recombinação e mutação originais do PSAR, além do método de cálculo de compatibilidade.

Nessa aplicação serão substituídos os operadores de recombinação e mutação por operadores clássicos utilizados em diversas aplicações do Algoritmo Genético (HOLLAND, 1975). Para a recombinação será utilizado o método clássico de recombinação por um ponto, assim como apresentado na Seção 3.2.1.1 relativo ao método 1PX, que utiliza um ponto de corte entre dois cromossomos para a realização da troca do material genético dos cromossomos envolvidos. O posicionamento do ponto de corte se dará a partir do hiperparâmetro *taxa de recombinação* definido antes da inicialização do algoritmo. A mutação irá introduzir uma alteração aleatória em cada gene do cromossomo dada uma probabilidade, essa probabilidade se tornará o hiperparâmetro *taxa de mutação* do modelo. O cálculo de compatibilidade e seleção será o mesmo adotado no modelo BP_APF, da Seção anterior 4.5.2.1 que analisa posições correspondentes com valores iguais no cromossomo. Nele, quanto mais posições iguais entre os cromossomos, maior será a compatibilidade, que varia no intervalo [0 1], 0 para nenhum valor igual em

posições correspondentes no cromossomo, 1 para todas as posições iguais no cromossomo.

A partir do código conceitual do PSAR apresentado logo abaixo, já explicado com maiores detalhes na Seção 2.2, as alterações no cálculo da compatibilidade se concentram nas linhas 6 e 7 do código, onde se realiza o cálculo de compatibilidade do grupo de indivíduos S^t em relação a ele mesmo e dos grupos S^t e $S^{t\prime}$ em relação ao melhor indivíduo $best^t$ da geração *t*. A alteração ocorre também na linha 9 do código, onde ocorre a recombinação e mutação.

1: **procedimento** PSAR
2: Definir $t = 0$
3: Escolher $best^t$
4: **repetir**
5: Gerar S^t usando μ_t
6: Gerar $S^{t\prime}$ a partir da compatibilidade entre os indivíduos de S^t
7: Gerar $p_{selected}$ a partir da compatibilidade entre os indivíduos de S^t e $S^{t\prime}$ em relação ao $best^t$
8: Encontrar $best^t$ em S^t
9: Definir $best(S^t) = D(S^t, best^t) =$ $\begin{cases} p_r^t = (1 - \alpha\rho_r{}^t)s^t + (\alpha\rho_r{}^t)s^{t'} \\ p_m^t = best^t + \rho_m{}^t(p_{selected}^t - p_r^t) \\ best^t \end{cases}$
10: Definir $t = t + 1$
11: **até** critério de parada ser alcançado
12: **fim do procedimento**

4.5.4 Busca Participativa com Transferência Seletiva (BP_APF_TS)

Na Seção 1.3 é apresentada a Transferência Seletiva (TS) como uma alternativa de operador nos algoritmos participativos apresentados em (LIU, 2016), sendo um substituto do processo de recombinação. Essencialmente, a transferência seletiva é uma substituição filtrada entre cromossomos de uma sequência para outra, sem excluir a possibilidade de que toda a sequência seja copiada [BIRCHENHALL et al., 1997]. Em (LIU, 2016) são apresentadas 4 instâncias de algoritmos participativos, sendo que 2 utilizam a TS em seu processo, a PSST (*Participatory Search with Selective Transfer*) e PDST (*Differential Participatory Search with Selective Transfer*).

O modelo desenvolvido BP_APF_TS, pretende utilizar a estratégia do modelo BP_APF, apresentada na Seção 4.5.2 que reinterpreta o algoritmo PSAR, alterando somente o operador de recombinação pela TS e mantendo o método de cálculo de compatibilidade e mutação como apresentado na Seção 4.5.2.

4.6 Critério de parada

Dentro do método proposto foram criados três critérios de paradas. O primeiro é a parada pela quantidade total de gerações. Essa parada é acionada quando a quantidade de gerações passadas atinge um valor pré-estabelecido. Geralmente os outros critérios serão acionados antes deste, entretanto, ele é útil para limitar o tempo total de execução caso os outros critérios não sejam acionados.

O segundo critério é acionado se não houver uma melhora significativa da aptidão dentro de uma quantidade determinada de gerações passadas. Entretanto, deve-se levar em consideração a característica de neutralidade. Portanto, a quantidade de gerações passadas não deve ser muito limitada, pois pode haver genes inativos em um determinado momento que irão aumentar a aptidão de uma forma significativa no futuro.

O terceiro critério é acionado se a aptidão do conjunto de validação na geração atual for menor que a aptidão em um número estipulado de gerações passadas, ou seja, se houver uma queda consistente na aptidão da validação. É passado para a próxima geração o genótipo que representa a geração em que a aptidão do conjunto de validação atingiu o seu máximo.

5 Estudo de Casos

O estudo de casos foi dividido em duas etapas. Na primeira, referente à Seção 5.1, foram utilizadas base de dados artificiais para definir alguns parâmetros do modelo BP_APF. Esses parâmetros incluem tanto parâmetros referentes ao método de busca quanto às características das APF. O método BP_APF proposto possui dois conjuntos de parâmetros: o primeiro conjunto está relacionado diretamente com a representação utilizada, herdados da PGC, e ao método de busca participativa, dentre os quais podemos citar: número de indivíduos na população, número de nós, número de gerações, função de aptidão, *levelback,* W_1 e W_2. O segundo conjunto de parâmetros está ligado à árvore propriamente dita, sendo eles: o número de partições *fuzzy* e os operadores *fuzzy*.

Na segunda etapa, referente à Seção 5.2, foram utilizados alguns diferentes algoritmos de classificação conhecidos na literatura, sendo eles discriminados na Seção 5.2.2, para efeito comparativo em relação ao desempenho apresentado. Nessa segunda etapa foram utilizadas algumas bases de dados reais conhecidas que serão detalhadas na Seção 5.2.1. Na Seção 5.3 será apresentado um exemplo de aplicação que ilustra o processo de utilização das árvores em um caso real de classificação.

5.1 Definição de parâmetros e alguns resultados a partir de base de dados artificiais

Nesta primeira etapa serão definidos alguns parâmetros do método proposto e será verificada sua estabilidade do processo evolutivo, utilizando algumas bases de dados artificiais. Nessas bases de dados artificiais o número de pontos disponíveis fica à critério do usuário, o que torna esses dados atraentes na definição de parâmetros uma vez que o estudo não fica limitado a uma quantidade máxima de amostras. As diversas bases de dados artificiais disponíveis possibilitam o uso de diferentes tipos de distribuição de probabilidades, o que permite a análise do modelo em diversos cenários sem a limitação que dados reais estabelecem. As bases de dados artificiais utilizadas nesse estudo de casos já foram utilizadas em trabalhos anteriores tais como em (KRIJTHE; HO; LOOG, 2012), (BREIMAN, 1996), (LOCAREKJUNGE; WEIHS, 2010), (SCHIFFNER, 2010), (ERTUĞRUL; KAYA, 2014) para avaliar algoritmos de aprendizado. A Tabela 12 aponta

quais as bases de dados artificiais serão usadas nessa primeira etapa do estudo de casos, indicando também a quantidade de pontos, atributos e classes de cada uma.

Tabela 12 – Bases de dados artificiais usadas na primeira etapa do estudo de casos

Base de dados	Pontos	Atributos	Classes
Conjunto Banana	1000	2	2
Highleyman	900	2	2
Lithuanian	900	2	2
Difficult_2D	1200	2	2
Difficult_6D	1200	6	2
Difficult_10D	2000	10	2
Difficult_20D	2400	20	2

Em (SANTOS, 2014) diversos estudos foram realizados com êxito, tendo como objetivo determinar os melhores parâmetros utilizados no algoritmo, baseado na PGC, para síntese das APF. Esse estudo foi realizado de forma que o método desenvolvido obtivesse o resultado esperado, unindo níveis competitivos de acurácia na tarefa de classificação com uma boa interpretabilidade. Baseado nos parâmetros alcançados nesses estudos, foi realizado uma série de experimentos com o objetivo de encontrar os melhores parâmetros (parâmetros *default*) para o método aqui proposto. Partindo dessa premissa, os parâmetros foram fixados nos melhores parâmetros encontrados em (SANTOS, 2014) e a partir de uma série de experimentos serão feitas variações dos parâmetros de forma que os melhores sejam escolhidos. Os parâmetros iniciais foram fixados da seguinte forma: Nós = 400; $Levelback$ = 399; Tamanho da população = 50; Quantidade de gerações = 300; Partições = 5; Função de aptidão = Similaridade a partir do RMSE; conjunto reduzido de operadores *fuzzy*; $W_1 = 0{,}7$ e $W_2 = 0{,}3$.

Cada experimento terá como objetivo a determinação de um parâmetro em específico. As técnicas usadas nos experimentos serão: validação cruzada (VC) com 5 repetições, utilizando as mesmas partições em todos os casos; curvas de aprendizado com 30 repetições (CA); *Repeated Hold Out* com 10 repetições para gerar as curvas de aprendizado. A Tabela 13 mostra um resumo da sequência de experimentos que serão realizados, assim como o parâmetro a ser definido e a técnica utilizada.

Tabela 13 – Sequência de experimentos na definição dos parâmetros *default*

Experimento	Parâmetro a ser definido	Técnica
1°	-	CA
2°	Operadores *fuzzy*	VC
3°	Número de partições *fuzzy*	VC
4°	W1 e W2	VC
5°	Função de aptidão	VC
6°	-	Repeated hold out e curvas de evolução

No primeiro experimento foram traçadas as curvas de aprendizagem para cada base de dados. As curvas de aprendizado (*Learning Curves*) têm como principal objetivo estimar uma taxa de erro empírica baseada na quantidade de amostras do conjunto de dados para treinamento (MUKHERJEE et al., 2003). A curva de aprendizagem permite estabelecer se existe algum tipo de erro no processo de aprendizagem, de forma que é possível detectar erros de *bias* e variância. O erro de bias captura a ideia de que o modelo faz suposições errôneas sobre os dados, um exemplo disso é quando um algoritmo aprende uma função linear quando a função a ser aprendida é senoidal, o que gera um erro sistemático. O erro de variância captura as alterações de desempenho de um algoritmo em função de pequenas variações do conjunto de treinamento. As curvas de aprendizagem que serão apresentadas nos experimentos mostram, para cada base de dados, o erro de classificação em relação ao conjunto de teste e o erro de treinamento em relação ao conjunto de treinamento, ambos considerando a variação do tamanho do conjunto de treino ao longo do gráfico. Para se obter resultados mais estáveis foram feitas trinta repetições para cada base de dados. A Figura 38 apresenta um resumo das curvas de aprendizado geradas a partir de cada base de dados. Nela é apresentada a proporção máxima e mínima em relação ao tamanho do grupo de treinamento em que cada base de dados atingiu estabilidade. As curvas de aprendizado são apresentadas com detalhes nas Figura 41, Figura 42, Figura 43, Figura 44, Figura 45, Figura 46 e Figura 47 do Apêndice B.

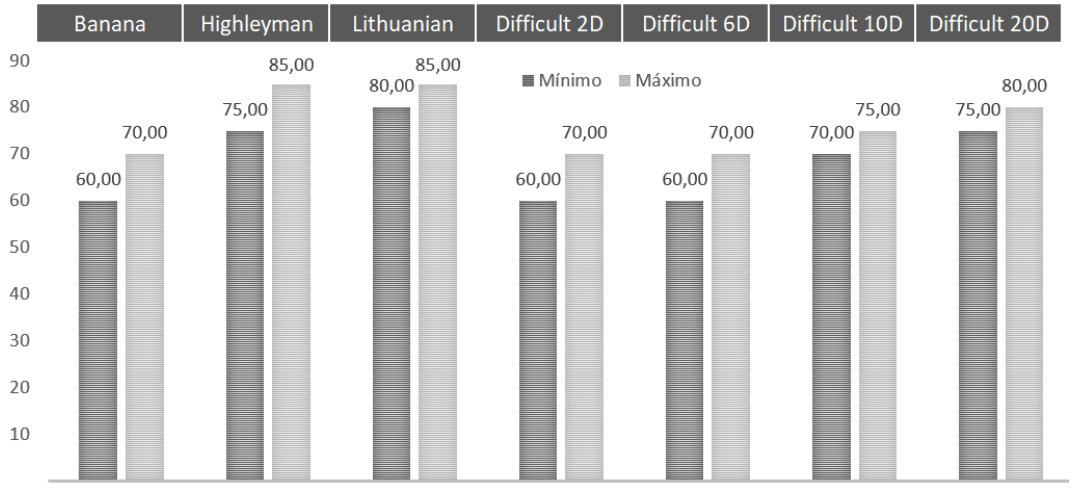

Figura 38 – Proporção máxima e mínima (em %) em relaçao ao tamanho do grupo de treinamento em que cada base de dados atingiu estabilidade nas curvas de aprendizado

Observando as curvas de aprendizado geradas é possível notar uma maior variação da relação entre os erros de teste e treinamento quando o conjunto de treinamento possui um número reduzido de pontos. Ao longo dos gráficos é possível notar a convergência dos erros e uma redução nessa variação, indicando uma tendência de estabilidade com o aumento do tamanho do conjunto de treinamento. Essa estabilidade em relação aos erros de teste e treinamento indicam que o algoritmo consegue aprender sem que ocorra um treinamento excessivo, o que poderia ocasionar problemas de generalização. Em média, os erros de teste e treinamento se aproximam e tendem a uma estabilidade quando se utiliza entre 65% a 75% do total de pontos disponíveis para o conjunto de treinamento. A análise das curvas de aprendizado aponta a inexistência de erros de bias ou variância que comprometam o algoritmo.

No segundo experimento foi feita a variação da quantidade de operadores utilizados. Em um cenário são utilizados todos os operadores da APF original. Em um segundo cenário é utilizado um conjunto reduzido de operadores, este conjunto foi descrito na Seção 2.3. Como medidas de desempenho foram utilizadas a acurácia e o AUC, que é a área sob a curva ROC (FAWCETT, 2006).

Os resultados apresentados na Tabela 14 indicam que o modelo com os operadores escolhidos apresenta uma discreta vantagem em relação ao modelo com todos os operadores. A realização do teste de Friedman (DERRAC et al. 2011) aliado ao teste de comparação múltipla Nemenyi para um grau de confiança de 95% ($\alpha = 0{,}05$), indica que não existe diferença estatisticamente significativa entre os resultados obtidos. No modelo

desenvolvido foi definido utilizar o conjunto reduzido de operadores. Um conjunto menor de operadores, na representação das árvores, contribui para uma melhor compreensão da expressão que a árvore representa.

Tabela 14 – Resultados do segundo experimento da primeira etapa do estudo de casos

Base de dados	Todos os operadores		Operadores escolhidos	
	Acurácia	AUC	Acurácia	AUC
Banana	0,8181±0,0284	**0,900 ±0,0235**	**0,8285±0,0278**	0,8971±0,0418
Highleyman	0,7591±0,1609	0,650±0,1064	**0,8013±0,1399**	**0,9014±0,1119**
Lithuanian	0,7851±0,0657	0,7316±0,0654	**0,9098±0,0363**	**0,9159±0,0417**
Difficult_2D	0,6920±0,0548	0,7526±0.0693	**0,6940±0,0726**	**0,7671±0.0437**
Difficult_6D	**0,6590±0,0753**	0,7428±0,0610	0,6562±0,0502	**0,7476±0,0706**
Difficult_10D	0,6698±0,0664	**0,7495±0,0659**	**0,6865±0,0708**	0,7475±0,0736
Difficult_20D	**0,6165±0,0646**	0,6852±0,0932	0,6021±0,0672	**0,6904±0,0891**

No terceiro experimento desenvolvido, foi feita a variação das partições *fuzzy*. Foram criados três cenários, com três, cinco e sete partições. Os resultados são apresentados na Tabela 15.

Tabela 15 – Resultados do terceiro experimento da primeira etapa do estudo de casos

Base de dados	3 Divisões		5 Divisões		7 Divisões	
	Acurácia	AUC	Acurácia	AUC	Acurácia	AUC
Banana	0,79±0,03	0,90±0,02	**0,82±0,0363**	**0,91±0,02**	0,82±0,02	0,89±0,02
Highleyman	0,70±0,1	**0,91±0,03**	**0,82±0,0727**	0,90±0,07	0,76±0,13	0,85±0,13
Lithuanian	0,81±0,07	0,93±0,05	**0,89±0,0443**	**0,93±0,03**	0,78±0,05	0,85±0,05
Difficult_2D	0,66±0.08	**0,75±0,10**	**0,68 ±0,06**	0,73±0.07	0,62±0.05	0.68±0.06
Difficult_6D	0,64±0,06	**0,70±0,09**	**0,68±0,06**	0,64±0,07	0,57±0,06	0,59±0,07
Difficult_10D	0,64±0,0	**0,65±0,09**	**0,67±0,04**	0,63±0,07	0,57±0,06	0,58±0,05
Difficult_20D	**0,61±0,6**	**0,61±0,07**	0,61±0,06	0,56±0,07	0,54±0,05	0,50±0,07

A partir dos resultados apresentados na Tabela 15, foi realizado o teste Friedman, aliado ao teste de comparação múltipla Nemenyi para um grau de confiança de 95%, e não foi observada diferença estatisticamente significativa entre os métodos. Optou-se por utilizar o modelo com cinco partições, mantendo os parâmetros iniciais estabelecidos no início dessa etapa do estudo de casos.

A altura da árvore é um dos fatores que influem na complexidade da expressão representada. Sendo assim, no quarto experimento, avaliou-se o desempenho do sistema com a variação dos pesos das duas parcelas (W_1 *e* W_2) da função de aptidão, explicados na seção 4.4. O peso W_1 é relativo à medida de similaridade no treinamento e W_2 é relativo ao tamanho da árvore. O campo profundidade, apresentado nas Tabela 16 e Tabela 17, indica a profundidade média da árvore de cada classe, ou seja, a distância em níveis, entre a raiz da árvore a folha mais distante, indicando o quanto o método é interpretável. Dessa forma, quanto menor a profundidade, maior a interpretabilidade. É apresentado também o erro médio e a área sob a curva ROC.

Tabela 16 – Resultados do quarto experimento para definição dos parâmetros *default*
Parte 1

Base de dados	$W_1 = 1$ / $W_2 = 0$			$W_1 = 0.7$ / $W_2 = 0.3$		
	Profundidade	Acurácia	AUC	Profundidade	Acurácia	AUC
Banana	[5,44 5,72]	**0,89±0,02**	**0,93±0,02**	[3,36 3,04]	0,87±0,02	0,91±0,03
Highleyman	[6,64 6,92]	0,72±0,13	**0,76±0,15**	[2,6 2,36]	**0,81±0,13**	0,74±0,15
Lithuanian	[5,067 5,8]	**0,91±0,02**	**0,9±0,02**	[3,333 2,2]	0,89±0,03	0,90±0,05
Difficult_2D	[6,4 6,25]	**0,67±0,06**	**0,75±0,06**	[1,5 1,93]	0,67±0,06	0,73±0,07
Difficult_6D	[6,76 6,32]	**0.69±0.07**	**0,75±0.07**	[2,52 2,52]	0,68±0.07	0.72±0.08
Difficult_10D	[7,48 7,32]	**0.66±0.06**	0,72±0.09	[2,6 2,96]	0,68±0.07	**0.73±0.09**
Difficult_20D	[7,44 7]	**0.66±0.06**	**0,74±0.07**	[2,92 2,8]	0,66±0.07	0.71±0.08

Tabela 17 - Resultados do quarto experimento para definição dos parâmetros *default* - Parte 2

Base de dados	$W_1 = 0.5$ / $W_2 = 0.5$		
	Profundidade	Acurácia	AUC
Banana	**[2,32 2,28]**	0,84±0,02	0,86±0,03
Highleyman	**[1,92 1,68]**	0,69±0,13	0,72±0,14
Lithuanian	**[1,53 1,93]**	0,87±0,02	0,87±0,03
Difficult_2D	**[1,27 1,24]**	0,644±0,06	0,69±0,07
Difficult_6D	**[1,56 1,56]**	0,65±0.06	0,71±0.08
Difficult_10D	**[1,76 1,92]**	0,65± 0.07	0,70±0.09
Difficult_20D	**[2,08 1,88]**	0,64±0.08	0,71±0.09

É possível notar a relação antagônica que existe entre acurácia e interpretabilidade analisando a variação existente nos resultados apresentados nas Tabela 16 e Tabela 17. Com a mudança dos parâmetros W_1 e W_2 proposta, nota-se uma redução de desempenho com o aumento da limitação do tamanho da árvore. Optou-se pela relação $W_1 = 0{,}7$ e $W_2 = 0{,}3$, pois apresentam uma relação equilibrada entre a profundidade (interpretabilidade do modelo) e acurácia. Nessa relação mantém-se um bom resultado e ainda se limita o crescimento da árvore.

No quinto experimento foi feita a variação da função utilizada para definir a aptidão durante o treinamento. As funções utilizadas foram a "Similaridade a partir do RMSE", a "AUC" e a "Entropia Cruzada". Os resultados são apresentados na Tabela 18.

Tabela 18 - Resultados do quinto experimento da primeira etapa do estudo de casos

Base de dados	Similaridade a partir do RMSE			AUC			Entropia Cruzada		
	Profundidade	Acurácia	AUC	Profundidade	Acurácia	AUC	Profundidade	Acurácia	AUC
Banana	**[3 2,36]**	0,85± 0,03	**0,86± 0,03**	[3,72 4,4]	**0,87±0, 07**	0,86± 0,08	[2,96 3,04]	0,84± 0,0	0,85± 0,02
Highley man	**[1,56 1,48]**	**0,74± 0,10**	0,75± 0,12	[3,84 3,52]	0,66±0, 11	0,69± 0,14	[2,72 1,6]	0,73± 0,10	**0,76± 0,1**
Lithuania n	**[2,29 2,4]**	**0,88± 0,02**	0,90± 0,04	[3,28 2,92]	0,88±0, 04	**0,95± 0,03**	[3,52 2,64]	0,88± 0,03	0,91± 0,04
Difficult _2D	**[2,92 2,2]**	**0,67± 0,08**	**0,73± 0,09**	[4,84 4,72]	0,61±0, 091	0,68± 0,08	[4,44 3,68]	0,65± 0,06	0,72± 0,08
Difficult _6D	**[2,56 2,12]**	**0,69± 0,05**	**0,75± 0,06**	[5,68 4,28]	0,60±0, 07	0,67± 0,09	[4,04 3,68]	0,65± 0,07	0,71± 0,09
Difficult _10D	**[2,92 3,36]**	**0,66± 0,07**	0,72± 0,06	[4,88 5,28]	0,62±0, 07	0,70± 0,06	[5,28 4,88]	0,66± 0,07	**0,73± 0,09**
Difficult _20D	**[2,88 3]**	**0,65± 0,07**	0,7±0, 08	[5,48 5,16]	0,61±0, 08	0,69± 0,08	[4,72 4,16]	0,65± 0,06	**0,75± 0,0**

Os resultados apresentados na Tabela 18 indicam que o modelo com medida de similaridade a partir do RMSE obteve por mais vezes os melhores resultados, optando-se por esta função para definir a aptidão.

O sexto experimento foi realizado para verificar a capacidade de o método gerar árvores que possuíssem desempenho similar quando se utiliza os mesmos parâmetros e os mesmos conjuntos de treinamento e teste. Para que isso fosse possível foi utilizado o *Repeated Hold Out* com dez repetições ao invés da validação cruzada com cinco pastas. A Figura 39 apresenta o valor médio dos erros máximo, médio e mínimo quando

alcançam estabilidade ao longo das gerações para cada classe da base de dados, ou seja, o erro quando param e evoluir. As Figura 48, Figura 49, Figura 50 e Figura 51 do apêndice B mostram as curvas de evolução das classes das bases de dados Banana e Difficult_2D.

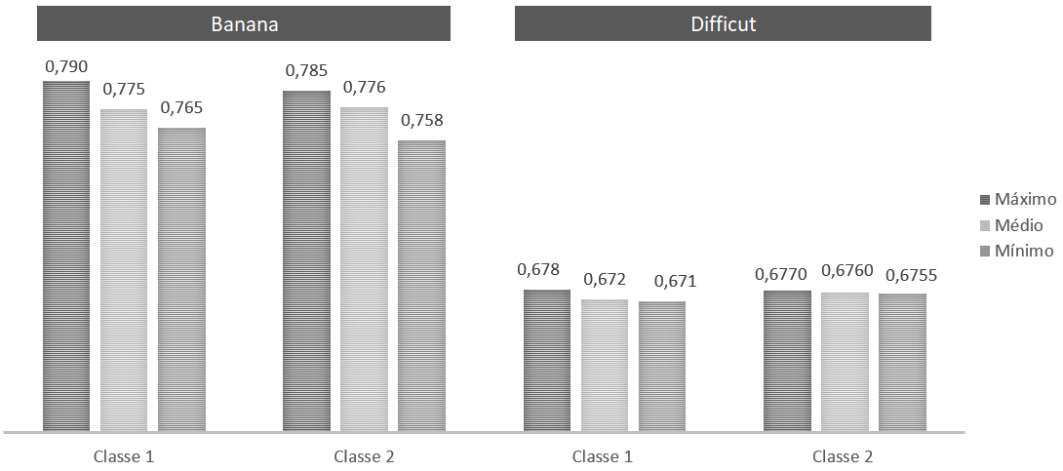

Figura 39 – Média dos erros máximo, médio e mínimo quando alcançam estabilidade ao longo das gerações das curvas de evolução

É observável que os valores estão próximos, denotando uma consistência do processo evolutivo.

Ao fim desta primeira etapa do estudo de casos, a configuração padrão de parâmetros se definem conforme apresentado na Tabela 19. Os parâmetros que não foram abordados nesta etapa foram definidos pelos mesmo apresentados no início desta seção.

Tabela 19 – Lista de parâmetros padrão (*default*) definidos na primeira etapa do estudo de casos

Parâmetro	Valor padrão
Número de gerações	300
Número de indivíduos	50
Quantidade de linhas	1
Quantidade de nós	400
Levelback	399
Conjunto de operadores *fuzzy*	Conjunto reduzido
Número de partições *fuzzy*	5
Pesos das parcelas da função de aptidão	$W_1 = 0{,}7$ / $W_2 = 0{,}3$
Função de aptidão	Similaridade a partir do RMSE

Duas variantes do BP_APF serão utilizadas nos testes com base de dados reais, a variante BP_APF1, que possui genótipos com 400 nós evoluídos em 300 gerações, resultante do estudo apresentado nesta seção. E a variante BP_APF2 que possui genótipos com 50 nós evoluídos em 1250 gerações, que tem como objetivo gerar árvores menores devido à quantidade reduzida de nós disponíveis para o processo evolutivo.

Nas implementações dos modelos BP_APF_ST, BP_APF_AP e BP_APF_AG serão utilizados os mesmos hiperparâmetros da versão BP_APF1. Porém a versão BP_APF_AG inclui dois novos a se definirem, relativos aos operadores de crossover e mutação, sendo estabelecidos, respectivamente, em 0,4 e 0,1.

5.2 Resultados obtidos

Nesta seção serão apresentados resultados experimentais utilizando bases de dados reais em diversos tipos de classificadores, inclusive com todos os modelos propostos. O objetivo é comparar os resultados dos métodos desenvolvidos com outros classificadores conceituados.

Todos os resultados nesta seção foram obtidos através da técnica de validação cruzada com 5 repetições. As medidas de desempenho utilizadas foram a acurácia e a área sob a curva ROC (FAWCETT, 2006). A área sob a curva ROC é um indicador importante porque nos fornece uma medida da precisão total independente de um limiar particular, porém ele se restringe em relação à quantidade de classes do problema. Dessa forma para todas as bases de dados com mais de 1 classe, a AUC é encontrada com base na média das AUCs calculadas a partir da estratégia “um contra todos”.

5.2.1 Conjunto de dados

Para testar e comparar o algoritmo foram utilizadas bases de dados que estão disponíveis no UCI Machine Learning Repository, no STATLIB e no LIBSVM. A Tabela 20 apresenta todas as bases de dados utilizadas e suas respectivas quantidades de pontos, atributos e classes.

Tabela 20 – Base de dados reais utilizadas

Base de dados	Pontos	Atributos	Classes
Iris	150	4	3
Wine	178	13	3
Sonar	208	60	2
Pima	768	8	2
Balance	625	4	3
Haberman	306	3	2
Lupus	87	3	2
BreastCancer	683	9	2
Australian	690	14	2
AnalcatdataLawsuit	264	4	2
Ionosphere	351	33	2
Bupa	345	6	2
Transfusion	378	4	2

5.2.2 Algoritmos classificadores usados para comparação

Para uma avaliação do desempenho do método desenvolvido com o estado da arte, foram usados os seguintes classificadores: Support Vector Machine Linear (SVM-L) (VAPNIK, 1999); K-nearest Neighbors (KNN) (WEBB, 2002); Random Forests (RF) (BREIMAN, 2001); Support Vector Machine Radial (SVM-R) (VAPNIK, 1999); APF1 e APF2 (SANTOS, 2014). Os algoritmos escolhidos são conhecidos por terem um bom desempenho em um grande número de benchmarks (CARUANA; NICULESCU-MIZIL, 2006).

Máquinas de Vetores de Suporte (*Support vector machines* - SVM) são algoritmos de aprendizado supervisionado usados com sucesso tanto em problemas de classificação quanto de regressão (BYUN; LEE, 2002), (SAPANKEVYCH; SANKAR, 2009). São sistemas baseados na teoria do aprendizado estatístico (HASTIE; TIBSHIRANI; FRIEDMAN, 2011). Para um problema de classificação com duas classes, a forma básica do SVM é o classificador linear. O SVM linear faz a classificação construindo um hiperplano que irá separar as classes de uma forma ótima. O hiperplano ótimo é o que cria uma margem máxima. A margem é definida como a distância de uma amostra do conjunto de treinamento e o hiperplano. Pode ser provado que esta solução particular tem a maior capacidade de generalização. Esta formulação pode ser generalizada pela aplicação de um mapeamento não linear no conjunto de treino. Os dados são transpostos para um novo espaço com muitas dimensões, onde as classes são separadas mais

facilmente e um hiperplano ótimo pode ser encontrado. É frequentemente usada para realizar este mapeamento não linear uma função de base radial, sendo frequentemente o primeiro mapeamento não linear a se considerar. Embora a superfície de decisão (hiperplano) seja linear no espaço com muitas dimensões, quando observado no espaço original ele não é mais linear, indicando que o SVM pode ser aplicado também para dados que não são linearmente separáveis. (GOLDBAUM et al., 2002).

O KNN é um algoritmo de aprendizado não paramétrico usado para classificação, ou seja, não faz suposições sobre a distribuição dos dados nem sobre a função discriminante que gera, de forma que a estrutura do modelo é determinada a partir dos dados de treinamento. Ele apresenta alguns problemas com variância e se torna sensível à *outliers* quando se utiliza um número baixo de vizinhos próximos no seu treinamento. Sua finalidade é usar um banco de dados no qual os pontos de dados são separados em várias classes para prever a classificação de um novo ponto de amostragem. O algoritmo KNN propõe um aprendizado baseado em instâncias, ou aprendizado preguiçoso, significando que na fase de aprendizagem, ele simplesmente armazena um conjunto de instâncias rotuladas (conjunto de treinamento). Quando um novo ponto tem que ser classificado, o algoritmo encontra uma quantidade K de instâncias dentro do conjunto de treinamento perto do ponto que se deseja classificar, usando uma função de similaridade geralmente baseada na distância euclidiana. A classificação é realizada em favor da classe que é mais representada no conjunto dos K objetos mais próximos. Se K = 1, então o objeto é simplesmente atribuído à classe de seu vizinho mais próximo (AMARAL, et al. 2013).

O algoritmo Florestas Aleatórias (*Random Forests* - RF) é um método de aprendizado de máquina flexível e fácil de usar que produz um ótimo resultado na maioria das vezes. É também um dos algoritmos mais utilizados porque é simples e pode ser usado para tarefas de classificação e regressão. É um é um método de aprendizado que produz e combina diversas árvores de decisão (BREIMAN, 2001). Se usado para classificação, sua saída será a classe que possui um maior número de árvores de decisão apontando como correta, ao passo que se for utilizado para a regressão, apresenta a média dos resultados das árvores individuais. O algoritmo foi desenvolvido por Breiman (2001) apresentando dois aspectos chave: A ideia de "bagging" criada por Breiman e a seleção aleatória de características (HO, 1998), (AMIT; GEMAN, 1997). O uso de bagging ajuda a reduzir a variância, fazendo a média de muitos modelos ruidosos, porém aproximadamente não tendenciosos. Árvores são candidatas ideais para o bagging, pois

elas podem capturar nos dados, estruturas com interação complexa que tenham erro médio baixo. (HASTIE; TIBSHIRANI; FRIEDMAN, 2011).

A seleção de características aleatória auxilia a redução da variância do *bagging*, reduzindo a correlação entres as árvores. Isto é realizado no processo de crescimento da árvore, através da seleção aleatória das variáveis de entrada. Particularmente, no processo de crescimento de uma árvore individual em uma base de dados "bootstrapped", antes de cada divisão, um subconjunto de $m \leq p$ entre as p variáveis de entrada, é selecionado de forma aleatória para serem os candidatos a calcular a melhor divisão do conjunto de treinamento. O classificador RF é rápido e apresenta um desempenho no nível do estado da arte (CARUANA; NICULESCU-MIZIL, 2006). Ele pode lidar com uma grande quantidade de variáveis de entrada e oferece uma estimação interna do erro de generalização durante o processo de criação das árvores. Outra característica importante é a habilidade de ranquear a importância das variáveis, aparentemente para medir a força de predição de cada variável. Também podendo computar proximidade entre os objetos, o que é útil para formação de cluster, detectar limites e visualizar os dados (pela escala) (SANTOS, 2014).

5.2.3 Resultados

Nesta seção serão apresentados alguns estudos experimentais realizados para se obter uma ideia do desempenho dos modelos criados comparando com os outros classificadores citados na seção 5.2.2 e utilizando as bases de dados citadas na seção 5.2.1.

Os hiperparâmetros dos outros classificadores foram obtidos da seguinte forma: O número de vizinhos K foi fixado em 1; o número de árvores geradas no classificador RF foi igual a 50 e o número de atributos sob o qual é feita a partição foi igual a 1. Os parâmetros de regularização do SVM de base linear e de base radial foram determinados a partir de busca pelo melhor desempenho em uma validação cruzada interna. Para cada partição, foi feita uma validação cruzada que utilizava apenas o conjunto de treinamento da respectiva partição. Este mesmo método foi utilizado para encontrar o valor do parâmetro *r* que define a base radial. Esta forma de encontrar os parâmetros é baseada na validação cruzada aninhada (CAWLEY; TALBOT, 2010) é tida como uma forma de evitar o *overfit*.

As Tabela 21 e Tabela 22 apresentam o desempenho dos algoritmos com relação a acurácia e o AUC respectivamente. A Tabela 23 apresenta o *rank* médio obtido por cada

algoritmo. Em todas a tabelas do estudo de casos, os melhores resultados encontram-se destacados.

Tabela 21 – Acurácia

Base de dados	RF	KNN	SVM-L	SVM-R	APF1	APF2	BP_APF1	BP_APF2	BP_APF_ST	BP_APF_AP	BP_APF_AG
Iris	0,955±0,03	0,959±0,03	**0,977±0,02**	0,96±0,03	0,963±0,01	0,96±0,03	0,93±0,04	0,95±0,01	0,92±0,01	0,86±0,11	0,895±0,02
Wine	**0,98±0,02**	0,74±0,07	0,95±0,03	0,80±0,03	0,96±0,02	0,95±0,02	0,90±0,06	0,95±0,02	0,92±0,01	0,82±0,14	0,95±0,01
Sonar	0,851±0,05	0,853±0,05	0,812±0,04	**0,87±0,05**	0,763±0,03	0,76±0,03	0,696±0,06	0,761±0,02	0,806±0,02	0,69±0,07	0,805±0,03
Pima	0,75±0,02	0,67±0,03	0,76±0,03	0,76±0,03	0,76±0,01	0,76±0,01	0,72±0,02	0,74±0,01	0,75±0,01	0,71±0,05	**0,76±0,004**
Balance	0,85±0,03	0,64±0,03	0,880,01	**0,90±0,04**	0,76±0,05	0,74±0,04	0,79±0,04	0,72±0,04	0,83±0,02	0,54±0,11	0,77±0,03
Haberman	0,725±0,04	0,714±0,05	0,723±0,02	0,722±0,04	0,757±0,02	0,755±0,02	0,72±0,04	0,75±0,01	**0,76±0,01**	0,74±0,01	0,75±0,01
Lupus	0,63±0,13	0,68±0,09	0,74±0,07	0,77±0,07	0,77±0,02	0,77±0,02	0,74±0,08	**0,78±0,02**	0,77±0,02	0,76±0,05	0,77±0,02
Breast Cancer	0,968±0,02	0,957±0,01	0,967±0,01	0,965±0,03	0,957±0,01	0,957±0,01	0,958±0,02	0,96±0,01	0,97±0,01	0,85±0,09	**0,97±0,001**
Australian	0,84±0,02	0,65±0,03	0,85±0,02	0,69±0,04	0,85±0,01	0,85±0,01	0,827±0,07	0,855±0,01	**0,86±0,004**	0.83±0,08	0,855±0,01
Analcatdata Lawsuit	0.977±0.02	0.973±0.02	**0,984±0,02**	0.974±0.02	0.96±0.001	0.96±0.001	0.947±0.02	0.95±0.008	0.96±0.01	0,92±0,001	0.96±0.01
Ionosphere	0,9±0,04	0,859±0,03	0,87±0,03	**0,933±0,02**	0,89±0,02	0,87±0,03	0,87±0,04	0,84±0,02	0,854±0,02	0,80±0,05	0,87±0,02
Bupa	0,717±0,06	0,617±0,06	0,694±0,05	**0,727±0,05**	0,638±0,03	0,637±0,04	0,644±0,05	0,677±0,02	0,645±0,01	0,61±0,08	0,663±0,01
Transfusion	0,769±0.02	0,725±0.02	0,76±0.001	**0,780±0,02**	0,77±0.005	0,766±005	0,762±0.01	0,7672±0.005	0,77±0.003	0,74±0,08	0,77±0.003

Tabela 22 – AUC

Base de dados	RF	KNN	SVM-L	SVM-R	APF1	APF2	BP_APF1	BP_APF2	BP_APF_ST	BP_APF_AP	BP_APF_AG
Iris	0,99±0,006	0,98±0,01	0,97±0,05	0,99±0,02	0,99±0,01	0,99±0,01	0,97±0,004	0,99±0,005	**0,998±0,003**	0,91±0,11	0,998±0,01
Wine	**0,99±0,001**	0,82±0,03	0,96±0,02	0,94±0,04	0,99±0,01	0,98±0,02	0,96±0,02	0,94±0,03	0,94±0,03	0,83±0,13	**0,99±0,004**
Sonar	0,925±0,04	**0,945±0,03**	0,893±0,05	0,944±0,04	0,79±0,07	0,77±0,07	0,748±0,08	0,711±0,08	0,839±0,06	0,63±0,11	0,83±0,08
Pima	**0,81±0,02**	0,70±0,04	0,83±0,02	0,81±0,03	0,78±0,05	0,78±0,05	0,77±0,04	0,75±0,05	0,80±0,03	0,67±0,1	0,78±0,04
Balance	0,87±0,02	0,67±0,03	**0,97±0,01**	0,93±0,04	0,74±0,05	0,72±0,04	0,76±0,04	0,68±0,04	0,78±0,02	0,64±0,04	0,73±0,03
Haberman	0,656±0,07	0,608±0,05	**0,7±0,07**	0,6±0,07	0,67±0,09	0,67±0,07	0,63±0,07	0,627±0,06	0,66±0,07	0,63±0,07	0,67±0,08
Lupus	0,72±0,12	0,65±0,14	0,83±0,08	0,79±0,08	0,82±0,09	0,82±0,08	0,81±0,08	0,79±0,07	0,82±0,08	0,77±0,1	**0,83±0,07**
Breast Cancer	0,989±0,01	**0,99±0,01**	0,99±0,002	0,986±0,02	0,98±0,01	0,98±0,01	0,97±0,01	0,95±0,02	0,984±0,01	0,81±0,14	0,97±0,01
Australian	0,916±0,02	0,701±0,04	**0,923±0,02**	0,793±0,05	0,898±0,02	0,895±0,02	0,9±0,02	0,9±0,02	0,9±0,03	0,86±0,08	0,9±0,02
Analcatdata Lawsuit	**0,99±0,02**	0,983±0.02	0,995±0.01	0,976±0.04	0,97±0.05	0,98±0.025	0,94±0.08	0,946±0.06	0,926±0.09	0,89±0,18	0.985±0.01
Ionosphere	0,97±0,02	0,97±0,02	0,88±0,05	**0,98±0,02**	0,9±0,05	0,88±0,08	0,89±0,05	0,78±0,06	0,83±0,06	0,76±0,06	0,87±0,04
Bupa	0,761±0,06	0,63±0,07	0,717±0,06	**0,764±0,05**	0,617±0,09	0,611±0,09	0,664±0,06	0,651±0,06	0,662±0,05	0,61±0,08	0,647±0,05
Transfusion	0,71±0.05	0,616±0.05	**0,742±0,04**	0,696±0.05	0,631±0.09	0,674±0.07	0,712±0.04	0,714±0.04	0,719±0.04	0,67±0,08	0,701±0.04

Foi aplicado o teste de Friedman aliado ao teste de comparação múltipla Nemenyi para um grau de confiança de 95%, ($\alpha = 0{,}05$) nos resultados apresentados nas Tabela 21 e Tabela 22, e o *rank* médio desse teste apresentado na Tabela 23. As Tabela 24 e Tabela 25 apresentam os resultados de comparação múltipla Nemenyi feito 2 a 2, que indica entre quais algoritmos existe diferença estatisticamente significativa. Na Tabela 24 e Tabela 25 os valores "0" indicam que não existe diferença estatística. A partir da análise dos resultados, em relação à acurácia e à AUC, o método BP_APF_AP apresenta o pior resultado no *rank* médio apresentado na Tabela 23 enquanto o SVM-L apresenta o melhor.

Tabela 23 – Rank médio

Rank Médio	RF	KNN	SVM-L	SVM-R	APF1	APF2	BP_APF1	BP_APF2	BP_APF_ST	BP_APF_AP	BP_APF_AG
Acurácia	3,615	7,538	**2,884**	5,115	5,807	6,26	6,692	7,576	5,115	10,115	5,269
AUC	3,653	7,769	**2,961**	4,769	5,769	6,154	6,654	7,577	5,346	10,154	5,192

Tabela 24 – Teste de comparação múltipla com os resultados de Acurácia

	RF	KNN	SVM-L	SVM-R	APF1	APF2	BP_APF1	BP_APF2	BP_APF_ST	BP_APF_AP	BP_APF_AG
RF	0	1	0	0	0	0	0	0	0	1	0
KNN	-	0	1	1	1	0	0	0	0	0	1
SVM-L	-	-	0	0	0	0	0	0	0	1	0
SVM-R	-	-	-	0	0	0	1	0	0	1	0
APF1	-	-	-	-	0	0	0	0	0	1	0
APF2	-	-	-	-	-	0	0	0	0	1	0
BP_APF1	-	-	-	-	-	-	0	0	0	0	0
BP_APF2	-	-	-	-	-	-	-	0	0	0	0
BP_APF_ST	-	-	-	-	-	-	-	-	0	1	0
BP_APF_AP	-	-	-	-	-	-	-	-	-	0	1
BP_APF_AG	-	-	-	-	-	-	-	-	-	-	0

Tabela 25 – Teste de comparação múltipla com os resultados de AUC

	RF	KNN	SVM-L	SVM-R	APF1	APF2	BP_APF1	BP_APF2	BP_APF_ST	BP_APF_AP	BP_APF_AG
RF	0	0	0	0	0	0	0	0	0	1	0
KNN	-	0	1	0	0	0	0	0	0	0	0
SVM-L	-	-	0	0	0	0	0	1	0	1	0
SVM-R	-	-	-	0	0	0	0	0	0	1	0
APF1	-	-	-	-	0	0	0	0	0	1	0
APF2	-	-	-	-	-	0	0	0	0	0	0
BP_APF1	-	-	-	-	-	-	0	0	0	0	0
BP_APF2	-	-	-	-	-	-	-	0	0	0	0
BP_APF_ST	-	-	-	-	-	-	-	-	0	1	0
BP_APF_AP	-	-	-	-	-	-	-	-	-	0	1
BP_APF_AG	-	-	-	-	-	-	-	-	-	-	0

Na Tabela 26 é apresentado a profundidade alcançada por cada método de indução das APF. É apresentada a profundidade média de cada classe, ou seja, uma base de dados que possui 3 classes, como o caso do conjunto Iris, possui 3 valores referentes aos valores médios da árvore de cada classe.

Tabela 26 – Profundidade por cada classe das bases de dados

Base de dados	APF1	APF2	BP_APF1	BP_APF2	BP_APF_ST	BP_APF_AP	BP_APF_AG
Iris	[1,2 1,2 1,24]	[1,4 1 1]	[1,84 1,2 2,24]	**[1 1 1]**	[1,44 1 1]	[1,08 1,16 1,48]	**[1 1 1]**
Wine	[1 1,12 1]	[1,12 1,04 1,2]	[2,44 2,36 1,72]	[1,08 1,12 1,12]	[1,04 1.04 1]	[1,12 1,28 1,28]	**[1 1 1]**
Sonar	**[1,04 1]**	**[1 1,04]**	[2,64 2,68]	[1,08 1]	[1,52 1,88]	[1 1,08]	**[1 1,04]**
Pima	**[1 1]**	[1,04 1,04]	[3,08 3,52]	[1,04 1]	**[1 1]**	[1,04 1,16]	**[1 1]**
Balance	[1,36 1,48 1,4]	**[1,32 1,4 1,4]**	[1,68 4,92 5,32]	[1 2,36 2,4]	[1 1,96 1,92]	[2,24 1,96 2,12]	[1,16 2,04 2
Haberman	[1,04 1,08]	[1,04 1,08]	[3,56 3,32]	**[1 1]**	[2,6 2,2]	[1,2 1,24]	**[1 1]**
Lupus	**[1,08 1]**	[1 1,12]	[3,6 4,28]	[1,92 1,52]	[1.68 1.4]	[2,08 1,64]	[1.76 1.08]
Breast Cancer	**[1,04 1]**	**[1,04 1]**	[2,64 4,48]	[1,04 2,64]	[2,32 3,08]	[1,04 1,88]	[1 2,2]
Australian	[1 1,12]	[1,041, 16]	[1,92 2,96]	[1 1.12]	[1,72 2,24]	[1 1,16]	**[1 1]**
Analcatdata Lawsuit	[1 1,0]	**[1 1]**	[4,52 2,16]	[1.56 1.04]	[3.88 2.76]	[1,12 1,04]	[2,04 1]
Ionosphere	**[1 1,12]**	[1,08 1,16]	[2,52 4,4]	[1.08 1.2]	[1 1.64]	[1,12 1,08]	[1 1,36]
Bupa	[1,08 1]	[1,04 1]	[3,32 3,28]	**[1 1]**	[1,28 2,2]	[1,2 1,36]	**[1 1]**
Transfusion	[1,08 1,04]	[1,04 1]	[3,28 2,24]	[1.08 1]	[1,96 2]	[1,32 1,24]	**[1 1]**

Também foram comparados os modelos propostos com a versão original das APF apresentada anteriormente em (SENGE; HÜLLERMEIER, 2011). Da estratégia Top-Down criada por Senge e Hüllermeier foi utilizada para a comparação o PTTDE.25, que possui como parâmetros: beam = 5 e Parada Adaptativa = 0.25%. As características comparadas foram a acurácia e a profundidade média das árvores. A Tabela 27 apresenta a acurácia para cada algoritmo, a Tabela 28 apresenta a AUC e na Tabela 29 seus respectivos *ranks* médios.

Tabela 27 – Acurácia

Base de dados	**BP_APF 1**	**BP_APF 2**	**BP_APF _ST**	**BP_APF _AP**	**BP_APF _AG**	**PTTD. 25**
Iris	0,930	0,95	0,92	0,86	0,895	**0,960**
Wine	0,900	0,95	0,92	0,82	0,95	**0,977**
Sonar	0,696	0,761	**0,806**	0,69	0,805	0,689
Pima	0,72	0,74	0,75	0,71	**0,76**	**0,760**
Balance	0,79	0,72	0,83	0,54	0,77	**0,886**
Haberman	0,72	0,75	**0,76**	0,74	0,75	0,738
Lupus	0,74	**0,78**	0,77	0,76	0,77	0,770
Breast Cancer	0,958	0,96	**0,97**	0,856	**0,97**	0,713
Australian	0,827	0,855	**0,86**	0,83	0,855	0,854
Analcatdata Lawsuit	0.947	0.952	0.962	0,93	**0.965**	0,954
Ionosphere	0,87	0,84	0,854	0,80	0,87	**0,914**
Bupa	0,644	0,677	0,64	0,612	0,663	**0,707**
Transfusion	0,762	0,767	**0,77**	0,746	**0,77**	0,773

Tabela 28 - AUC

Base de dados	**BP_APF 1**	**BP_APF 2**	**BP_APF _ST**	**BP_APF _AP**	**BP_APF _AG**	**PTTD. 25**
Iris	0,97	**0,999**	0,998	0,903	0,998	0,997
Wine	0,96	0,94	0,94	0,83	**0,999**	0,997
Sonar	0,748	0,711	0,839	0,63	0,83	0,80
Pima	0,77	0,75	**0,80**	0,67	0,78	0,826
Balance	0,76	0,68	0,78	0,64	0,73	**0,944**
Haberman	0,63	0,627	0,66	0,63	**0,67**	0,659
Lupus	0,81	0,79	0,82	0,75	0,83	**0,838**
Breast Cancer	0,97	0,95	**0,984**	0,82	0,97	0,587
Australian	0,900	**0,922**	0,910	0,86	0,903	0,914
Analcatdata Lawsuit	0.94	0.946	0.926	0,89	**0.985**	0,984
Ionosphere	0,89	0,78	0,83	0,76	0,87	**0,953**
Bupa	0,664	0,651	0,662	0,612	0,647	**0,748**
Transfusion	0,712	0,714	0,719	0,675	0,701	**0,730**

Foi aplicado o teste de Friedman aliado ao teste de comparação múltipla Nemenyi, para um grau de confiança de 95%, nos resultados apresentados nas Tabela 27 e Tabela 28 e o *rank* médio deste teste apresentado na Tabela 29. A Tabela 30 e Tabela 31 apresentam os resultados de comparação múltipla Nemenyi feito 2 a 2, que indica entre quais algoritmos existe diferença estatisticamente significativa. Na Tabela 30 e Tabela 31 os valores "0" indicam que não existe diferença estatística. A partir da análise dos resultados em relação à acurácia e à AUC, foi constatada diferença estatisticamente significativa em relação ao método BP_APF_AG. O método BP_APF_AG apresenta o melhor *rank* médio em relação à acurácia, e *rank* médio em relação à AUC próximo ao melhor resultado do PTTD.25.

Tabela 29 – Rank médio

Rank Médio	**BP_APF1**	**BP_APF2**	**BP_APF_ST**	**BP_APF_AP**	**BP_APF_AG**	**PTTD.25**
Acurácia	4,5	3,1154	2,6154	5,5385	**2,5769**	2,6538
AUC	3,6154	3,9615	2,6923	5,8077	2,7692	**2,1538**

Tabela 30 – Teste de comparação múltipla com os resultados de Acurácia

	BP_APF1	**BP_APF2**	**BP_APF_ST**	**BP_APF_AP**	**BP_APF_AG**	**PTTD.25**
BP_APF1	0	0	0	0	0	0
BP_APF2	-	0	0	1	0	0
BP_APF_ST	-	-	0	1	0	0
BP_APF_AP	-	-	-	0	1	1
BP_APF_AG	-	-	-	-	0	0
PTTD.25	-	-	-	-	-	0

Tabela 31 – Teste de comparação múltipla com os resultados de AUC

	BP_APF1	**BP_APF2**	**BP_APF_ST**	**BP_APF_AP**	**BP_APF_AG**	**PTTD.25**
BP_APF1	0	0	0	1	0	0
BP_APF2	-	0	0	0	0	0
BP_APF_ST	-	-	0	1	0	0
BP_APF_AP	-	-	-	0	1	1
BP_APF_AG	-	-	-	-	0	0
PTTD.25	-	-	-	-	-	0

A segunda comparação foi feita com relação a profundidade média total das árvores, ou seja, a média em relação a todas as árvores criadas para cada base de dados. Os resultados são apresentados na Tabela 32, onde são comparados todos os algoritmos usados na síntese das APF.

Tabela 32 – Profundidade média total das árvores

Base de dados	**APF1**	**APF2**	**BP_APF1**	**BP_APF2**	**BP_APF_ST**	**BP_APF_AG**	**BP_APF_AP**	**PTTD.25**
Iris	1,213	1,133	1,76	**1**	1,147	**1**	1,24	3,667
Wine	1,040	1,120	2,173	1,107	1,027	**1**	1,22	6,667
Sonar	1,020	1,020	2,660	1,040	1,701	**1,013**	1,04	7
Pima	**1**	1,040	3,301	1,020	**1**	**1**	1,1	4
Balance	1,413	**1,373**	3,973	1,920	1,627	1,733	2,107	4,333
Haberman	1,060	1,060	3,440	**1**	2,410	**1**	1,22	3
Lupus	**1,04**	1,060	3,940	1,720	1,540	1,420	1,86	3
Breast Cancer	**1,021**	1,020	3,560	1,840	2,700	1,600	1,46	6
Australian	1,060	1,101	2,440	1,060	1,980	**1**	1,08	7
Analcatdata Lawsuit	**1**	**1**	3,340	1,300	3,320	1,020	1,08	6
Ionosphere	**1,060**	1,120	4,401	1,140	1,320	1,180	1,1	8
Bupa	1,040	1,020	3,301	**1**	1,740	**1**	1,28	5
Transfusion	1,060	1,020	2,760	1,040	1,980	**1**	1,28	5,33

A última comparação foi feita com relação ao número de avaliações necessárias para se chegar aos resultados apresentados, utilizando as bases de dados reais. Esta etapa tem como objetivo avaliar o custo computacional em relação a todos os algoritmos de síntese das APF utilizados no estudo de casos. Essa comparação foi feita em um cenário em que os algoritmos abordados, chegassem ao pior caso possível, onde não ocorra uma parada antecipada. Os resultados são apresentados na Tabela 33.

Tabela 33 – Quantidade de avaliações

	APF2	BP_APF_AG	BP_APF_ST	BP_APF2	BP_APF_AP	PTTD.25
Iris	15k	375k	375k	375k	375k	33k
Wine	15k	375k	375k	375k	375k	340k
Sonar	10k	250k	250k	250k	250k	1000k
Pima	10k	250k	250k	250k	250k	24k
Balance	15k	375k	375k	375k	375k	4k
Haberman	10k	250k	250k	250k	250k	2,7k
Lupus	10k	250k	250k	250k	250k	22,5k
BreastCancer	10k	250k	250k	250k	250k	63k
Australian	10k	250k	250k	250k	250k	54k
Analcatdata Lawsuit	10k	250k	250k	250k	250k	519k
Ionosphere	10k	250k	250k	250k	250k	99k
Bupa	10k	250k	250k	250k	250k	18k
Transfusion	10k	250k	250k	250k	250k	12k

É possível observar que os algoritmos propostos neste trabalho apresentam maior número de avaliações em relação aos outros abordados. Isso se deve a natureza do processo de busca participativa, onde ocorre mais de uma avaliação a cada geração, tornando-o custoso computacionalmente. Outro fator que acentua o custo no processo participativo utilizado, é a quantidade de indivíduos envolvidos no processo evolutivo quando comparado ao método APF2 que utiliza a PGC. Uma característica que pode ser notada é que a quantidade de atributos da base de dados utilizada influencia diretamente na quantidade de avaliações. Porém no PTTD.25 o número de avaliações aumenta consideravelmente com uma explosão no número de possibilidades na estratégia Top-Down, o que não ocorre nos outros algoritmos.

5.3 Exemplo de aplicação

Nesta seção será apresentado um exemplo de aplicação a partir de árvores geradas após o processo evolutivo utilizando o algoritmo BP_APF2 com os mesmos hiperparâmetros já estabelecidos no estudo de casos. O exemplo tem como objetivo classificar uma amostra a partir de uma base de dados. Neste exemplo foi utilizada a base de dados Iris, também utilizada no estudo de casos.

O conjunto de dados Flor Iris ou o conjunto de dados Iris de Fisher é um conjunto de dados multivariado introduzido pelo estatístico e biólogo britânico Ronald Fisher em 1936 em seu artigo “O uso de múltiplas medições em problemas taxonômicos como um exemplo de análise discriminante linear”. O conjunto de dados consiste em 50 amostras de cada uma das três espécies de Iris (Iris Setosa, Iris Virginica e Iris Versicolor). Quatro características (atributos) foram medidas a partir de cada amostra: o comprimento e a largura das sépalas e pétalas, em centímetros.

Para gerar as APF foi utilizado a proporção de 70% do conjunto total de dados no processo evolutivo de treinamento. Neste caso o processo evolutivo é composto por três cromossomos, cada um referente a uma classe da base de dados. A Figura 40 apresenta uma visão geral da aplicação de uma amostra de teste a ser classificada. No exemplo é usada uma amostra que não participou do processo de evolutivo. A amostra possui 4 atributos que serão fuzzificados pelas 5 partições utilizadas, gerando um total de 20 termos *fuzzy,* sendo 5 para cada atributo. A Tabela 34 apresenta os valores dos atributos referentes à amostra utilizada, onde também é apresentado o valor dos termos *fuzzy* gerados pela fuzzificação.

Tabela 34 – Valores dos atributos e dos termos *fuzzy* utilizados no exemplo de aplicação

	Atributo 1					Atributo 2					Atributo 3					Atributo 4				
Valor do atributo	5,1					3,4					1,5					0,4				
Valores dos termos *fuzzy*	0	0,7	0,2	0	0	0	0	0,7	0,3	0	0,6	0,3	0	0	0	0,5	0,5	0	0	0
	Termo *fuzzy* 1	Termo *fuzzy* 2	Termo *fuzzy* 3	Termo *fuzzy* 4	Termo *fuzzy* 5	Termo *fuzzy* 1	Termo *fuzzy* 2	Termo *fuzzy* 3	Termo *fuzzy* 4	Termo *fuzzy* 5	Termo *fuzzy* 1	Termo *fuzzy* 2	Termo *fuzzy* 3	Termo *fuzzy* 4	Termo *fuzzy* 5	Termo *fuzzy* 1	Termo *fuzzy* 2	Termo *fuzzy* 3	Termo *fuzzy* 4	Termo *fuzzy* 5

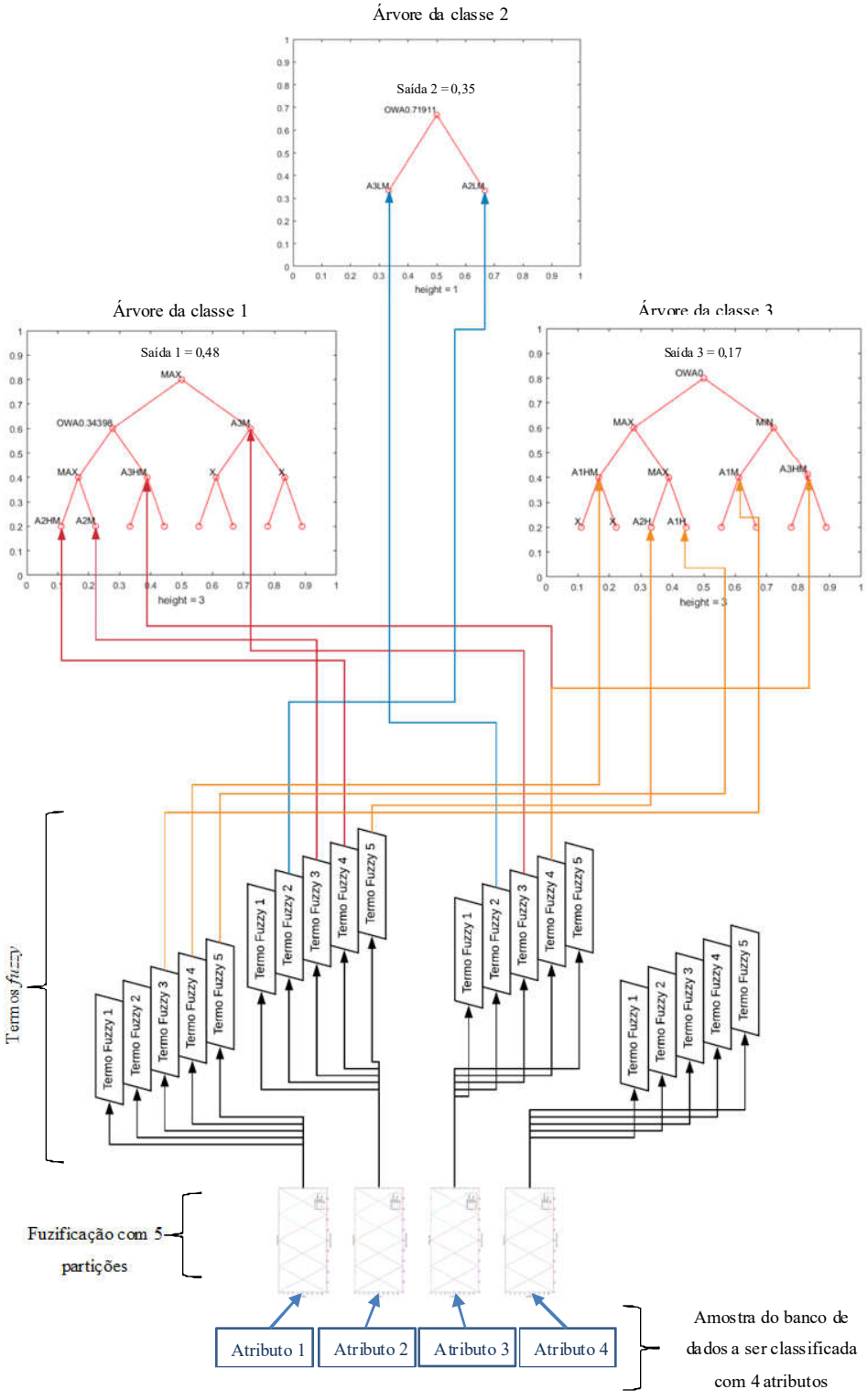

Figura 40 – Exemplo de aplicação das Árvores de Padrões Fuzzy

A Figura 40 indica a origem das entradas nas APF geradas de cada classe a partir dos termos fuzzificados, o que permite avaliar a importância de cada atributo no momento da classificação. Cada árvore gerada utiliza um conjunto pequeno de termos *fuzzy*, sendo esse conjunto de termos, os indicadores de quais atributos mais influenciaram na classificação. No exemplo as árvores das classes 1, 2 e 3 produziram, respectivamente, os valores: 0.48, 0.35 e 0.17, indicando que a amostra pertence à classe 1.

Os termos *fuzzy* gerados a partir do atributo de número 4 não são utilizados em nenhuma das árvores e serve como exemplo de atributo que tem baixa influência na classificação de uma amostra. Um exemplo de termo *fuzzy* que possui elevada relevância na classificação é o termo 4 do atributo 3. Ele está presente nas árvores das classes 1 e 3 de forma que uma alteração nesse termo afeta diretamente a saída de duas árvores simultaneamente. E ainda, este termo está no nível mais elevado da árvore da classe 1, indicando maior relevância uma vez que alterações no valor desse termo afeta de maneira mais acentuada o valor de saída da árvore de classe 1.

6 Conclusão

A capacidade de concentração, aprendizado e reconhecimento de padrões que a mente humana possui é única e desperta grande interesse em diversas áreas de estudos. Outro fator intrínseco à mente é a capacidade de armazenar informações e padrões aprendidos de forma que a informação que se encontra disponível seja analisada e compreendida, podendo ser generalizada. Porém, nos dias atuais, a imensa quantidade de informações disponíveis pelos mais diversos meios prejudica essa capacidade analítica. Diante desta dificuldade, áreas como a de mineração de dados e aprendizado de máquinas tentam extrair conhecimento de forma que possa ser possível apresentá-lo de maneira compreensível, como pelos Sistemas Fuzzy Baseados em Regras (SFBR). Modelos como o SFBR se tornam atraentes pois possuem uma capacidade de generalização competitiva, expressando o conhecimento adquirido através de estruturas qualitativas expressas em termos de linguagem natural.

A Árvore de Padrões Fuzzy (APF) é um método *fuzzy* hierárquico que surge com alternativa ao clássico SFBR. Modelos *fuzzy* hierárquicos possuem a característica de melhorar a relação antagônica que existe entre acurácia e interpretabilidade devido ao seu tipo de estrutura, além de reduzir o problema com a "maldição da dimensionalidade".

Este trabalho apresenta um método de indução de forma automática para o modelo hierárquico das APF na tarefa de classificação. O método de aprendizado das APF foi substituído pelo processo de busca participativa. No método participativo, a busca prossegue orientada pela compatibilidade entre indivíduos de uma população ao longo de gerações, sempre mantendo o melhor indivíduo nas populações posteriores e introduzindo indivíduos aleatoriamente em cada passo do algoritmo em um processo evolutivo. O método de síntese proposto tem finalidade de explorar melhor o espaço de busca, baseado no desempenho superior que a busca participativa apresenta em relação ao estado-da-arte, aprimorando fatores como a capacidade de generalização e interpretabilidade.

Ao longo deste trabalho foram apresentadas algumas alternativas na adaptação do modelo de busca participativa na síntese do modelo *fuzzy* das APF. Ao todo foram propostos 4 modelos como alternativa na utilização do método participativo. Em todos os modelos foram utilizados, para representação cromossômica das árvores, uma lista de números inteiros com restrições, enquadrando a heurística de busca dentro da classe de problemas combinatórios. A representação cromossômica utilizada foi baseada nos trabalhos (SANTOS, 2014) e (ALMEIDA, 2017), em que utilizam a Programação

Genética Cartesiana (PGC) na síntese das APF com sucesso. O primeiro modelo BP_APF_AG utiliza o Algoritmo Genético (AG) clássico, substituindo e etapa de seleção pelo procedimento baseado na compatibilidade entre os indivíduos de uma população, assim como utilizado na busca participativa. No modelo BP_APF, as adaptações propostas estão centradas na reinterpretação da compatibilidade, recombinação e mutação utilizados no algoritmo participativo intitulado Busca Participativa com Recombinação Aritmética (*Participatory Search with Arithmetical Recombination* – PSAR) apresentado em (LIU, 2016). O modelo BP_APF_TS utiliza toda a adaptação proposta no modelo BP_APF, porém, substituindo o processo de recombinação pela Transferência Seletiva (LIU, 2016). E o modelo BP_APF_AP utiliza o método numérico integral do PSAR na busca das APF. Para isso toda a estrutura do PSAR é utilizada sem alterações nos operadores de seleção, recombinação e mutação, porém na avaliação dos cromossomos, uma normalização e arredondamento dentro do intervalor proporcional na representação cromossômica das APF é feita de forma que o cromossomo gere as APF.

Foram realizados diversos estudos de casos para obter uma melhor compreensão do funcionamento dos métodos propostos, desenvolvendo estratégias para obter uma melhor relação entre generalização e interpretabilidade, avaliando também o desempenho dos classificadores *fuzzy* gerados, buscando a melhor configuração dos parâmetros envolvidos. Nos estudos realizados também foi realizada uma comparação com modelos clássicos para resolução de problemas de classificação, tais como: Support Vector Machine Linear (SVM-L) (VAPNIK, 1999); K-nearest Neighbors (KNN) (WEBB, 2002); Random Forest (RF) (BREIMAN, 2001); Support Vector Machine Radial (SVM-R) (VAPNIK, 1999). Foram avaliadas medidas de desempenho que envolvem acurácia, interpretabilidade e custo computacional durante o processo evolutivo.

Nos diversos estudos de casos realizados foi comprovado que todos os modelos propostos apresentam desempenho competitivo em relação aos modelos utilizados na comparação, não apresentando diferenças estatisticamente significativas. Na comparação com o modelo PTTD.25, originalmente proposto em (SENGE; HÜLLERMEIER, 2011) para indução das APF, os modelos aqui desenvolvidos apresentam desvantagens em relação ao nível de acurácia, porém os resultados não evidenciam diferenças estatisticamente significativas de acordo com o teste estatística não-paramétrico Friedman (DERRAC et al., 2011) aliado ao teste de comparação múltipla Nemenyi para um grau de confiança de 95%. Ainda em relação ao modelo PTTD.25, foi evidenciado uma melhora considerável em relação a todos os modelos desenvolvidos, principalmente

o modelo BP_APF_AG, em relação aos níveis de interpretabilidade, gerando-se árvores menores. Na comparação com os métodos APF1 e APF2, apresentados em (SANTOS, 2014), não ficam evidenciadas diferenças significativas em relação aos níveis de acurácia, porém o método BP_APF_AG apresentou melhora nos níveis de interpretabilidade, gerando árvores melhores.

Em um cenário em que os algoritmos de síntese das APF abordados nas comparações, chegassem ao pior caso possível, onde não ocorra uma parada antecipada, verificou-se que todos os métodos aqui propostos apresentam um custo computacional superior em relação as modelos APF1 e APF2. Em relação ao método PTTD.25, na maioria dos casos, também apresentaram um custo computacional mais elevado com exceção para algumas bases de dados utilizadas. Porém em nenhum dos casos a aplicabilidade de nenhum método ficou comprometida, pois mesmo o método BP_APF_AP apresentando o pior resultado com relação a todos os outros com diferença estatística apresentada, ele ainda mantém níveis de acurácia aceitáveis com uma boa relação de interpretabilidade quando comparado ao método PTTD.25.

Como possíveis trabalhos futuros no desenvolvimento desta pesquisa, pode-se apontar:

- Aperfeiçoar o mecanismo de avaliação multiobjetivo utilizando uma implementação de ranqueamento de soluções baseado no conceito de dominância de Pareto (ZHOU et al., 2011);
- Aperfeiçoar os parâmetros de excitação presentes no PSAR uma vez que originalmente são ajustados aleatoriamente a cada geração, não considerando o quanto uma nova população está próxima ao melhor indivíduo da população anterior. Uma taxa de excitação auto-ajustável implicaria no quanto os operadores de recombinação e mutação devem atuar na geração atual, direcionando o processo evolutivo de maneira mais eficiente;
- Estudar e explorar novas representações cromossômicas de forma que o algoritmo PSAR seja utilizado com o mínimo de alterações;
- Utilizar base dados com atributos categóricos e um conjunto *fuzzy* Singleton na etapa de fuzzificação dos atributos de entrada. Um conjunto fuzzy é chamado de singleton se seu suporte é um único ponto em U e com de grau de pertinência igual a 1, $\mu(x) = 1$.

7 Referências

ABRAHAM, A. Adaptation of Fuzzy Inference System Using Neural Learning. In: NEDJAH, N.; MOURELLE, L. DE M. (Eds.). Fuzzy Systems Engineering. Studies in Fuzziness and Soft Computing. [s.l.] Springer Berlin Heidelberg, 2005. p. 53–83.

AGBALI, Muna et al. Mate choice for nonadditive genetic benefits correlate with MHC dissimilarity in the rose bitterling (Rhodeus ocellatus). Evolution, v. 64, n. 6, p. 1683-1696, 2010.

AMIT, Y.; GEMAN, D. Shape Quantization and Recognition with Randomized Trees. Neural Comput., v. 9, n. 7, p. 1545–1588, out. 1997.

AIMIN ZHOU, BO-YANG QU, HUI LI, SHI-ZHENG ZHAO. Ponnuthurai Nagaratnam Suganthan, Qingfu Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art.Swarm and Evolutionary Computation, 1: 32-49, 2011.

AMARAL, J. L., LOPES, A. J., JANSEN, J. M., FARIA, A. C., & MELO, P. L.. An improved method of early diagnosis of smoking-induced respiratory changes using machine learning algorithms. Computer methods and programs in biomedicine, v. 112, n. 3, p. 441-454, 2013.

ALMEIDA, Cesar Eduardo Rodrigues Ferreira. APFR-PGC: Síntese de Árvores Padrões Fuzzy de Regressão via Programação Genética Cartesiana. [s.l: s.n.], 2017.

BARBOSA, Gustavo Eduardo Carnaval Sistemas Fuzzy Hierárquicos com Análise Espectral Aplicados à Classificação Supervisionada – Rio de Janeiro: UFRJ/COPPE, 2012.

BAYDOKHTY, Mohsen Ebrahimian; ZEYNAL, Hossein; ZARE, Assef. Nonlinear load-frequency control: An approach using optimized hierarchical fuzzy systems. In: Electrical Engineering (ICEE), 2016 24th Iranian Conference on. IEEE, 2016. p. 311-316.

BIRCHENHALL, Chris; KASTRINOS, Nikos; METCALFE, Stan. Genetic algorithms in evolutionary modelling. Journal of Evolutionary Economics, v. 7, n. 4, p. 375-393, 1997.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization overview and conceptual comparison. ACM Computing Surveys, v. 35, n. 3, p. 268–308, 2003. 33.

BREIMAN, L. et al. Classification and Regression Trees. Monterey: Wadsworth, 1984. ISBN 0-534-98053-8.

BREIMAN, L. [Bias, Variance, and] Arcing Classifiers. University of California at Berkeley, Berkeley, California: Statistics Department, University of California, Berkeley, fev. 1996. Disponível em: <http://statreports.lib.berkeley.edu/accessPages/460.html>.

BREIMAN, L. Random Forests. Machine Learning, v. 45, n. 1, p. 5–32, 1 out. 2001.

BUSCH, Jeremiah W. The evolution of self-compatibility in geographically peripheral populations of Leavenworthia alabamica (Brassicaceae). American Journal of Botany, v. 92, n. 9, p. 1503-1512, 2005.

BYUN, H.; LEE, S.-W. Applications of Support Vector Machines for Pattern Recognition: A SurveyProceedings of the First International Workshop on Pattern Recognition with Support Vector Machines. Anais...: SVM '02.London, UK, UK: Springer-Verlag, 2002Disponível em: <http://dl.acm.org/citation.cfm?id=647230.719394>.

CARUANA, R.; NICULESCU-MIZIL, A. An Empirical Comparison of Supervised Learning AlgorithmsProceedings of the 23rd International Conference on Machine Learning. Anais...: ICML '06.New York, NY, USA: ACM, 2006Disponível em: <http://doi.acm.org/10.1145/1143844.1143865>.

CASILLAS, J. Accuracy Improvements in Linguistic Fuzzy Modeling. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

CAWLEY, G. C.; TALBOT, N. L. C. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. J. Mach. Learn. Res., v. 11, p. 2079–2107, ago. 2010.

CHAKRABORTY U. K. Ed., Advances in Differential Evolution, ser. Studies in Computational Intelligence. Springer-Verlag, 2008, vol. 143.

CHANG, Chia-Wen; TAO, Chin-Wang. A simplified implementation of hierarchical fuzzy systems. Soft Computing, p. 1-11, 2018.

CORDÓN, Oscar et al. Evolutionary tuning and learning of fuzzy knowledge bases. Genetic fuzzy systems, v. 19, 2001.

CORDÓN, O. A Historical Review of Evolutionary Learning Methods for Mamdanitype Fuzzy Rule-based Systems: Designing Interpretable Genetic Fuzzy Systems. Int. J. Approx. Reasoning, v. 52, n. 6, p. 894–913, set. 2011.

CRUZ, A.V.A; Vellasco, M.B.R; Pacheco, A.C, "Quantum-Inspired Evolutionary Algorithm for Numerical Optimization", In Evolutionary Computation, 2006. CEC 2006. IEEE Congress, pages 2630–2637, July.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, v. 1, n. 1, p. 3–18, 2011.

DEUTSCH, D. "Quantum Theory, the Church-turing Principle and the universal quantum computer", Proceedings of the Royal Society, v. 400, pp, 97-117, 1985.

DUDA, R. O.; HART, P. E.; STORK, D. G. Pattern Classification. [s.l.] John Wiley & Sons, 2012.

ERTUĞRUL, Ömer Faruk; KAYA, Yılmaz. A detailed analysis on extreme learning machine and novel approaches based on ELM. American Journal of computer science and engineering, v. 1, n. 5, p. 43-50, 2014.

FAWCETT, T. An Introduction to ROC Analysis. Pattern Recogn. Lett., v. 27, n. 8, p. 861–874, jun. 2006.

FEOKTISTOV V. Differential Evolution: In Search of Solutions, 1st ed., ser. Springer Optimization and Its Applications. Springer, October 2006.

FERRI, C.; HERNÁNDEZ-ORALLO, J.; MODROIU, R. An experimental comparison of performance measures for classification. Pattern Recognition Letters, v. 30, n. 1, p. 27–38, 1 jan. 2009.

GACTO, M. J.; ALCALÁ, R.; HERRERA, F. Interpretability of Linguistic Fuzzy Rulebased Systems: An Overview of Interpretability Measures. Inf. Sci., v. 181, n. 20, p. 4340–4360, out. 2011

GOLDBAUM, M. H. et al. Comparing machine learning classifiers for diagnosing glaucoma from standard automated perimetry. Investigative Ophthalmology & Visual Science, v. 43, n. 1, p. 162–169, jan. 2002.

GONCALVES, L. B. et al. Inverted Hierarchical Neuro-fuzzy BSP System: A Novel Neuro-fuzzy Model for Pattern Classification and Rule Extraction in Databases. Trans. Sys. Man Cyber Part C, v. 36, n. 2, p. 236–248, mar. 2006.

HAN, K.; Kim, J.H. “Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem”, in Proceedings of the 2000 Congress on Evolutionary Computation, pp. 1354-1360, July.

HAN, K.H.; Han, J.H. “Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization”, IEEE Transactions on Evolutionary Computation 2002, 6(6):580–593, December.

HAN, J., KAMBER, M. Data Mining: Concepts and Techniques. 2 ed. San Francisco, Morgan Kauffman Publishers, 2006.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. 2nd ed. 2009. Corr. 7th printing 2013 edition ed. New York, NY: Springer, 2011.

HERRERA, Francisco. Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence, v. 1, n. 1, p. 27-46, 2008.

HO, T. K. The Random Subspace Method for Constructing Decision Forests. IEEE Trans. Pattern Anal. Mach. Intell., v. 20, n. 8, p. 832–844, ago. 1998.

HOLLAND, J.H. “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975.

HOLLAND, J.H. “Adaptation in Natural and Artificial Systems”, 2nd edition, The MIT Press, 1992.

HUANG, Z.; GEDEON, T. D.; NIKRAVESH, M. Pattern Trees Induction: A New Machine Learning Method. IEEE Transactions on Fuzzy Systems, v. 16, n. 4, p. 958–970, ago. 2008.

HÜLLERMEIER, E. Fuzzy Methods in Machine Learning and Data Mining: Status and Prospects. Fuzzy Sets Syst., v. 156, n. 3, p. 387–406, dez. 2005.

ISHIBASHI, Rogério. EXTRACAO DE CONHECIMENTOS COM INTERPRETABILIDADE AUMENTADA UTILIZANDO MODELAGEM FUZZY E OTIMIZACAO MULTI-OBJETIVO. 2013. Tese de Doutorado. Instituto Tecnológico de Aeronáutica.

KLEMENT, E. P.; MESIAR, R.; PAP, E. Triangular norms. [s.l.] Springer Science & Business Media, 2013. v. 8.

KARR, C. AI Expert. Genetic algorithms for fuzzy controllers, journal, v.6, n.2, p. 26–33, 1991.

KOZA, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. ISBN 0262111705. MIT Press, 1992.

KRIJTHE, J. H.; HO, T. K.; LOOG, M. Improving cross-validation based classifier selection using meta-learning2012 21st International Conference on Pattern Recognition (ICPR). Anais... In: 2012 21ST INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION (ICPR). nov. 2012

LARRANÃGA, P. el al. "Genetic Algorithms for the Travelling Salesman Problem: A review of representations and operators. *Articial Intelligence Review,* v. 13, p.129-170, 1999. Disponível em: <citesser.ist.psu/article/naga99genetic.html>

LAWLER, E. "The Traveling Salesman Problem: A Guilded Tour of Combinatorial Optimization. Wiley, New York. 1985.

LEGARDA, Oscar Hernan Samudio. RandomFIS: Um Sistema de Classificação Fuzzy para Problemas de Alta Dimensionalidade. 2016. Tese de Doutorado. PUC-Rio.

LIU, Yi Ling; GOMIDE, Fernando. Participatory genetic learning in fuzzy system modeling. In: Genetic and Evolutionary Fuzzy Systems (GEFS), 2013 IEEE International Workshop on. IEEE, 2013. p. 1-7.

LIU, Yi Ling; GOMIDE, Fernando. Evolutionary participatory learning in fuzzy systems modeling. In: Fuzzy Systems (FUZZ), 2013 IEEE International Conference on. IEEE, 2013. p. 1-8.

LIU, Y. L.; GOMIDE, F. Participatory search algorithms in fuzzy modeling. In: Proceedings of the World Conference in Soft Computing, Berkeley, USA. 2016.

LIU, Y. L. "Participatory Search Algorithms and Applications".– Campinas 2016.

LOCAREK-JUNGE, H.; WEIHS, C. Classification as a Tool for Research: Proceedings of the 11th IFCS Biennial Conference and 33rd Annual Conference of the Gesellschaft für Klassifikation e.V., Dresden, March 13-18, 2009. [s.l.] Springer, 2010.

MEZURA-MONTES E.; VELAZQUEZ-REYES J.; COELHO C. A. C. "A comparative study of differential evolution variants for global optimization," in Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO. ACM, 2006, pp. 485-492.

MILLER, J. F. Cartesian Genetic Programming. 2011 edition ed. Heidelberg : New York: Springer, 2011.

MILLER, J. F.; HARDING, S. L. Cartesian Genetic Programming Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. Anais...: GECCO '09.New York, NY, USA: ACM, 2009 Disponível em: <http://doi.acm.org/10.1145/1570256.1570428>.

MILLER, J. F.; THOMSON, P. Cartesian Genetic Programming. In: POLI, R. et al. (Eds.). Genetic Programming. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 2000. p. 121–132.

MUKHERJEE, S. et al. Estimating Dataset Size Requirements for Classifying DNA Microarray Data. Journal of Computational Biology, v. 10, n. 2, p. 119–142, 1 abr. 2003.

NAUCK, D.; KLAWONN, F.; KRUSE, R. Foundations of Neuro-Fuzzy Systems. New York, NY, USA: John Wiley & Sons, Inc., 1997.

NAVARRO, G. A Guided Tour to Approximate String Matching, ACM Computing Surveys, vol. 33, no. 1, pp. 32-88, Mar. 2001.

NEMHAUSER G. L.; WOLSEY L. A. Integer and Combinatorial Optimization. John WJ.ley, Chichester, UK, 1988.

NICOLAU, A. S. “Computação Quântica e Inteligência de Enxames Aplicados na Identificação de Acidentes de uma Usina Nuclear PWR” [s.l: s.n.], 2010.

ONWUBOLU G. C. "Scheduling flow shops using differential evolution algorithm," European Journal of Operational Research, vol. 171, no. 2, pp. 674-692, 2006.

PAN Q. K.; WANG L.; QIAN B. "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problem," Computers & Operations Research, vol. 36, no. 8, pp. 2498-2511, 2009.

PAPADIMITRIOU,C.H.; STEIGLITZ,K.Combinatorial optimization: algorithms and complexity. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN 0-13152462-3.

PATKI, Neha; WEDGE, Roy; VEERAMACHANENI, Kalyan. The synthetic data vault. In: Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on. IEEE, 2016. p. 399-410.

PEDRYCZ, W.; GOMIDE, F. An Introduction to Fuzzy Sets: Analysis and Design. Cambridge: NetLibrary, Incorporated, 1998. (Complex adaptive systems). ISBN 9780262161718.

PELTOKANGAS, R.; SORSA, A. Real-coded genetic algorithms and nonlinear parameter identification. 28 p. April 2008. ISBN 978-951-42-8785-5. ISBN 978-951-42-8786-2 (pdf).

PRICE, K. V.; STOM R. M. "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces;' Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, December 1997.

PRICE, K. V, "An introduction to differential evolution," in New Ideas in Optimisation, ser. Advanced Topics in Computer Science, D. Come, M. Dorigo, and F. Glover, Eds. McGraw-Hill, 1999, pp. 79-108.

PRICE, K. V.; STOM R. M.; LAMPINEN I.A. Differential Evolution: A Practical Approach to Global Optimization, 1st ed., ser. Natural Computing Series. Springer, December 2005.

PRADO R. S.; SILVA R. C. P.; GUIMARÂES F. G.; NETO O. M. "Using Differential Evolution for Combinatorial Optimization: A General Approach" GECCO CGECCO Comp '14 Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. Pages 69-70. Vancouver, BC, Canada — July 12 - 16, 2014.

ROISENBERG, Mauro et al. Emergência da inteligência em agentes autônomos através de modelos inspirados na natureza. Tese de Doutorado, Universidade Federal de Santa Catarina, Florianópolis, 1998.

ROSENDO, M. "Um Algoritmo de Otimização por Núvem de Partículas para Resolução de Problemas Combinatórios". [s.l: s.n.], 2010.

SANTOS, A. R.; DO AMARAL, J. L. M. Síntese de Árvores de padrões Fuzzy Através da Programação Genética Cartesiana. [s.l: s.n.], 2014.

SAPANKEVYCH, N. I.; SANKAR, R. Time Series Prediction Using Support Vector Machines: A Survey. Comp. Intell. Mag., v. 4, n. 2, p. 24–38, maio 2009.

SCHMITZ, G.; ALDRICH, C.; GOUWS, F. Knowledge-based Neurocomputing. Cambridge: MIT Press, 2000. 369–402 p. ISBN 0-262-03274-0. Dispon´ıvel em: . Acesso em: 16 Dez. 2013.

SENGE, R.; HÜLLERMEIER, E. Top-Down Induction of Fuzzy Pattern Trees. Trans. Fuz Sys., v. 19, n. 2, p. 241–252, abr. 2011.

SCHIFFNER, J. B. B. Bias-Variance Analysis of Local Classification Methods. p. 49– 57, 2010.

SILVEIRA, L. R.; TANSCHEIT R.; VELLASCO M. “Algoritmo Genético com Inspiração Quântica Aplicados a Problemas de Otimização Combinatória de Ordem” X SBAI – Simpósio Brasileiro de Automação Inteligente, vol X, pp. 779-784, setembro de 2011.

SILVEIRA, L. R. “Algoritmo Genético de Ordem com Inspiração Quântica” [s.l: s.n.], 2014.

SOLIS, Francisco J.; WETS, Roger J.-B. Minimization by random search techniques. Mathematics of operations research, v. 6, n. 1, p. 19-30, 1981. YAGER, R. A model of participatory learning. Man and Cybernetics, IEEE Transactions on Systems, 20(5):1229–1234, 1990.

SOUZA, F. J.; VELLASCO, M. M. R.; PACHECO, M. A. C. Hierarchical Neuro-fuzzy Quadtree Models. Fuzzy Sets Syst., v. 130, n. 2, p. 189–205, set. 2002.

STORN R.; PRICE K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11(4):341–359, 1997.

THRIFT, Philip R. Fuzzy Logic Synthesis with Genetic Algorithms. In: ICGA. 1991. p. 509-513.

TORRA, V. A review of the construction of hierarchical fuzzy systems. International Journal of Intelligent Systems, v. 17, n. 5, p. 531–543, 1 maio 2002.

VAPNIK, V. The Nature of Statistical Learning Theory. 2nd edition ed. New York: Springer, 1999.

WANG, D.; ZENG, X.-J.; KEANE, J. A. A Survey of Hierarchical Fuzzy Systems (Invited Paper). 2006.

WEBB, A. R. Density Estimation – Nonparametric. In: Statistical Pattern Recognition. [s.l.] John Wiley & Sons, Ltd, 2002. p. 81–122.

WITTEN, I. H.; FRANK, E. Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

YAGER, R. Participatory genetic algorithms. BISC Group List, 2000.

ZADEH, L. Fuzzy sets. Information and Control, v. 8, n. 3, p. 338 – 353, 1965. ISSN 0019-9958.

ZADEH, Lotfi A. Fuzzy sets. In: Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh. 1996. p. 394-432.

ZHANG, Xiao et al. Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction. Transportation Research Part C: Emerging Technologies, v. 43, p. 127-142, 2014.

ZHANG, W. Complete Anytime Beam SearchProceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. Anais...: AAAI '98/IAAI '98.Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998 Disponível em: <http://dl.acm.org/citation.cfm?id=295240.295709>.

ZHOU, A. et al. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation, v. 1, n. 1, p. 32–49, mar. 2011.

Apêndice A – Pseudocódigo do Algoritmo PSAR

1: **procedure** PSAR
2: f an objective function
3: $s \in S^t$ and $s' \in S^{t'}$
4: set *best* randomly
5: set $a_0 = 0$; $t \leftarrow 0$
6: **while** $t \leq t_{max}$ **do**
7: generate population S^t randomly
8: *last* $(S^t) \leftarrow best$
9: $S^{t'} \leftarrow s' = argmax_{r \in S^t}(\rho(s, r))$
10: find *best* in S^t
11: *Selection:*
12: compute $\rho^s(s, best)$ and $\rho^{s'}(s', best)$
13: **if** $\rho^s \geq \rho^{s'}$ **then**
14: $p_{selected} = s$
15: **else**
16: $p_{selected} = s'$
17: **end if**
18: *Recombination:*
19: choose $\alpha, \beta \in [0,1]$ randomly
20: compute $\rho_r = \rho(s, s')$
21: compute $a_{t+1} = a_t + \beta((1 - \rho_r) - a_t)$
22: $\rho_r = (1 - \alpha\rho^{1-a_{t+1}})s + \alpha\rho_r^{1-a_t}s'$
23: *Mutation:*
24: compute $\rho_m = \rho(p_{selected}, p_r)$
25: $p_m = best + \rho_m^{1-a_{t+1}}(p_{selected} - p_r)$
26: **if** $f(p_r)$ better than $f(best)$ **then**
27: $best \leftarrow p_r$
28: **end if**
29: **if** $f(p_r)$ better than $f(best)$ **then**
30: $best \leftarrow p_m$
31: **end if**
32: $t \leftarrow t + 1$
33: **end while**
34: return *best*
35: **end procedure**

Apêndice B – Curvas de Evolução e Aprendizado do Estudo de Casos

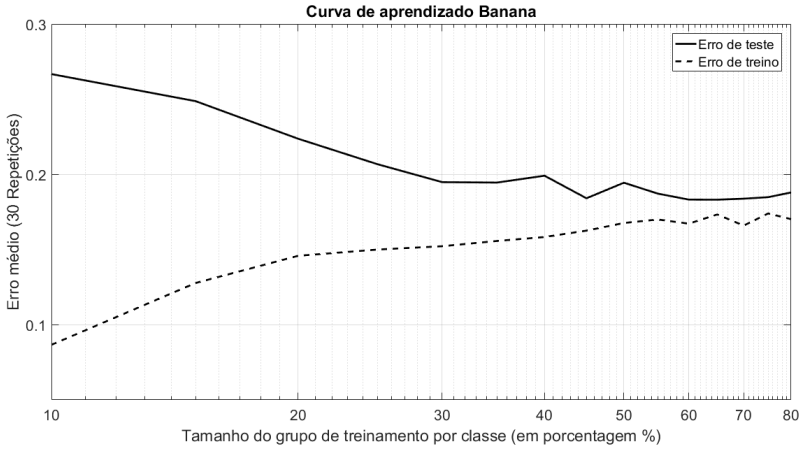

Figura 41 – Curva de aprendizado da base de dados Banana

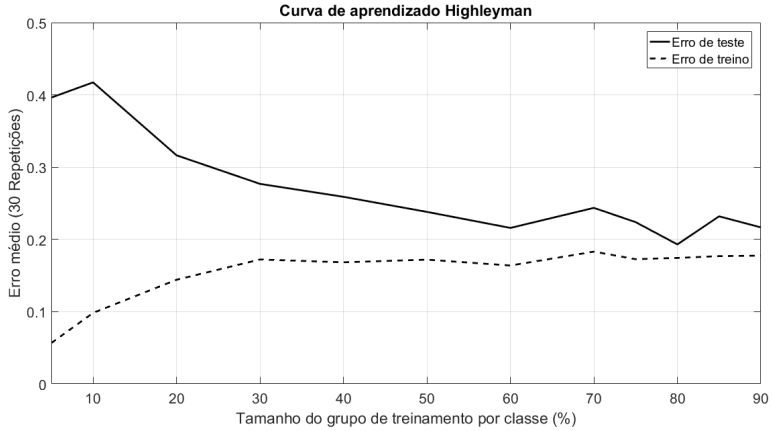

Figura 42 – Curva de aprendizado da base de dados Highleyman

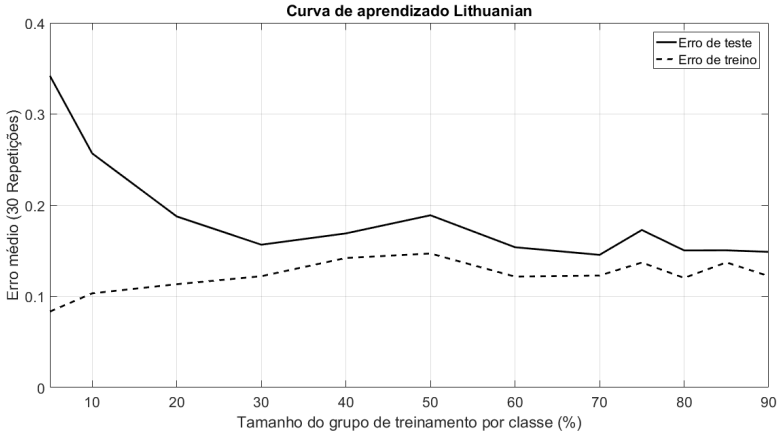

Figura 43 – Curva de aprendizado da base de dados Lithuanian

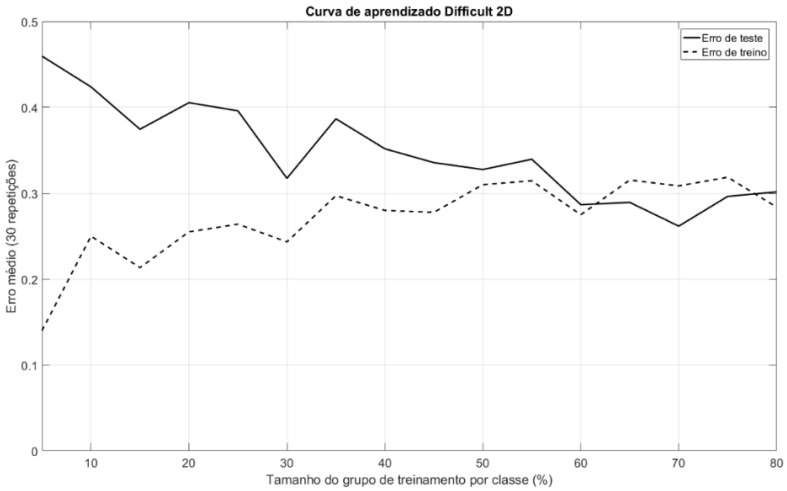

Figura 44 – Curva de aprendizado da base de dados Difficult 2D

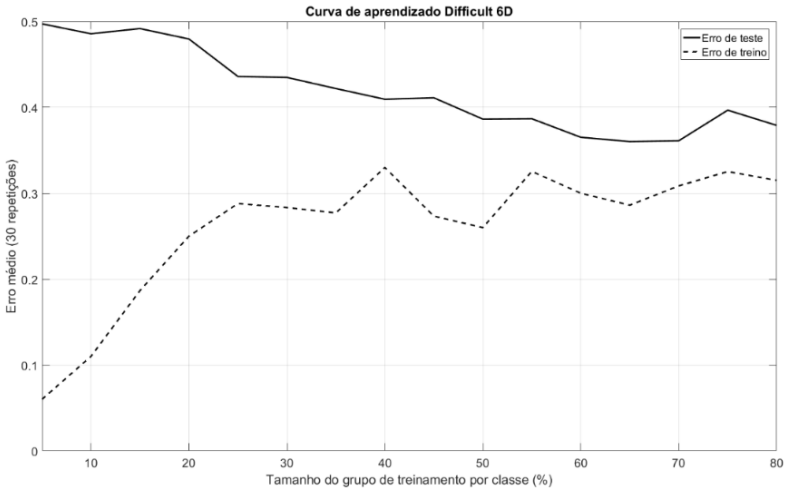

Figura 45 – Curva de aprendizado da base de dados Difficult 6D

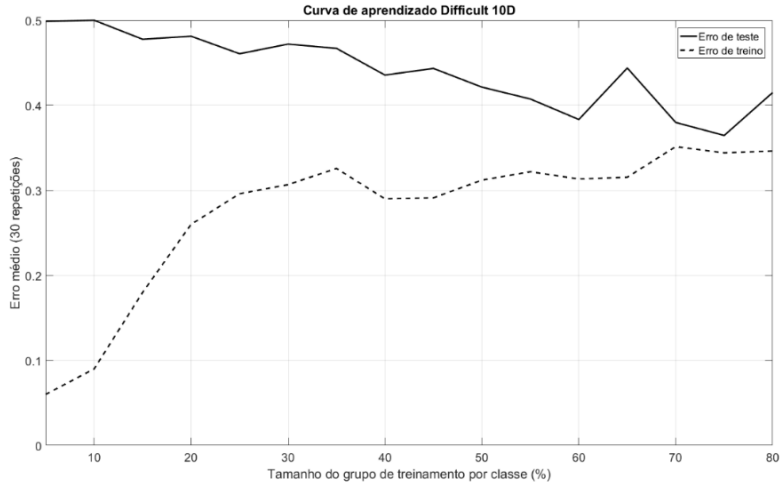

Figura 46 – Curva de aprendizado da base de dados Difficult 10D

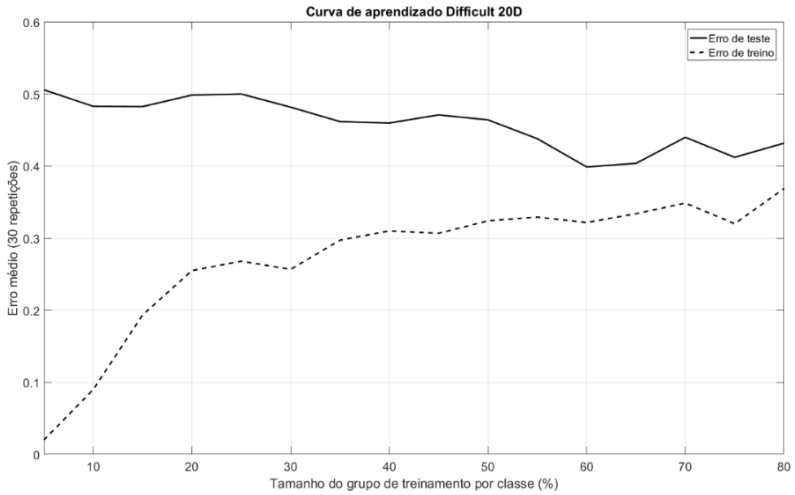

Figura 47 – Curva de aprendizado da base de dados Difficult 20D

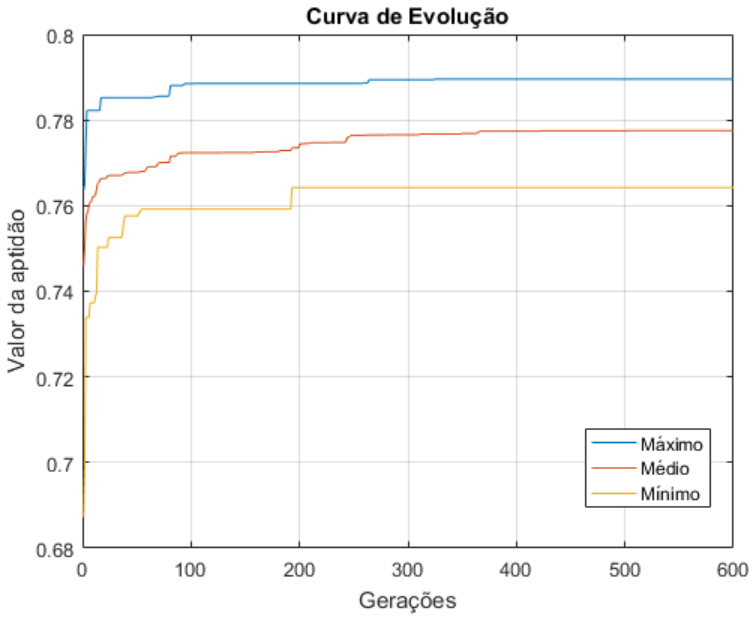

Figura 48 – Curva de evolução da base de dados Banana, classe 1

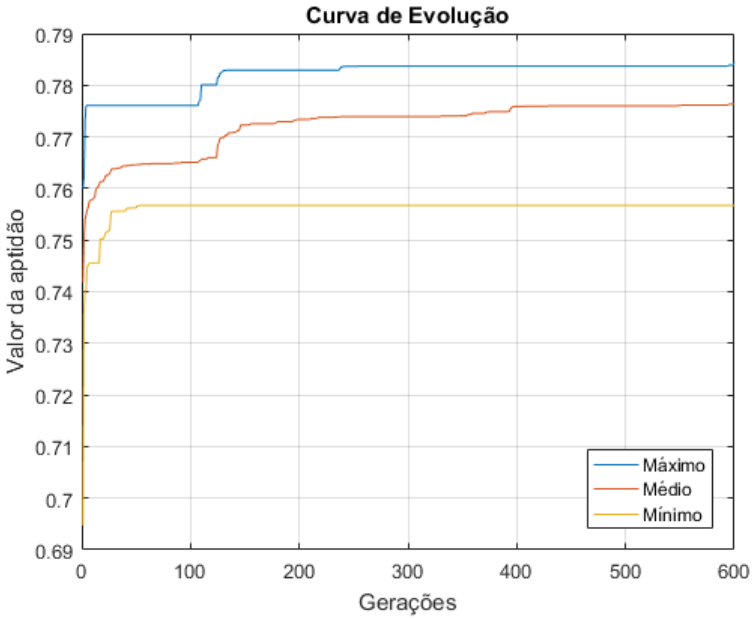

Figura 49 – Curva de evolução da base de dados Banana, classe 2

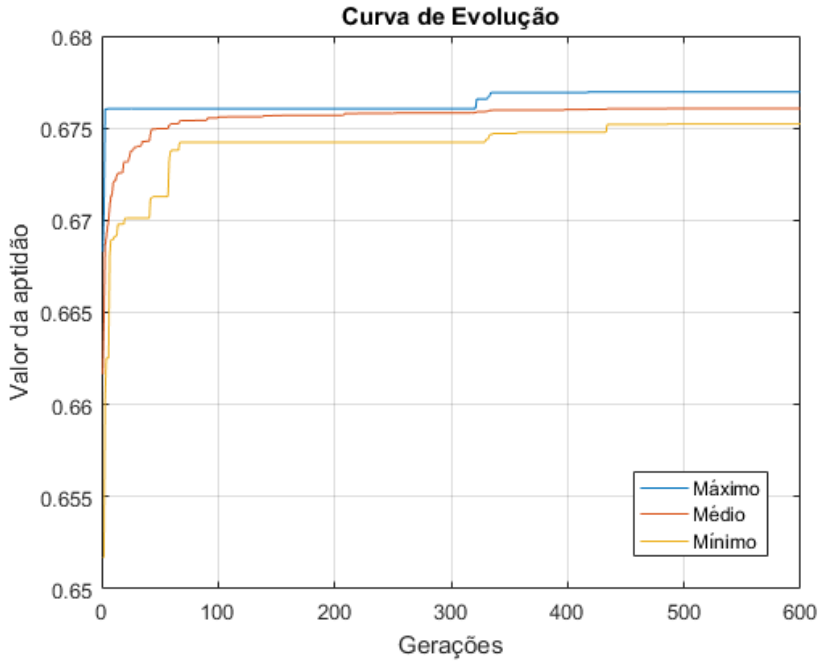

Figura 50 – Curva de evolução da base de dados Difficult2D, classe 1

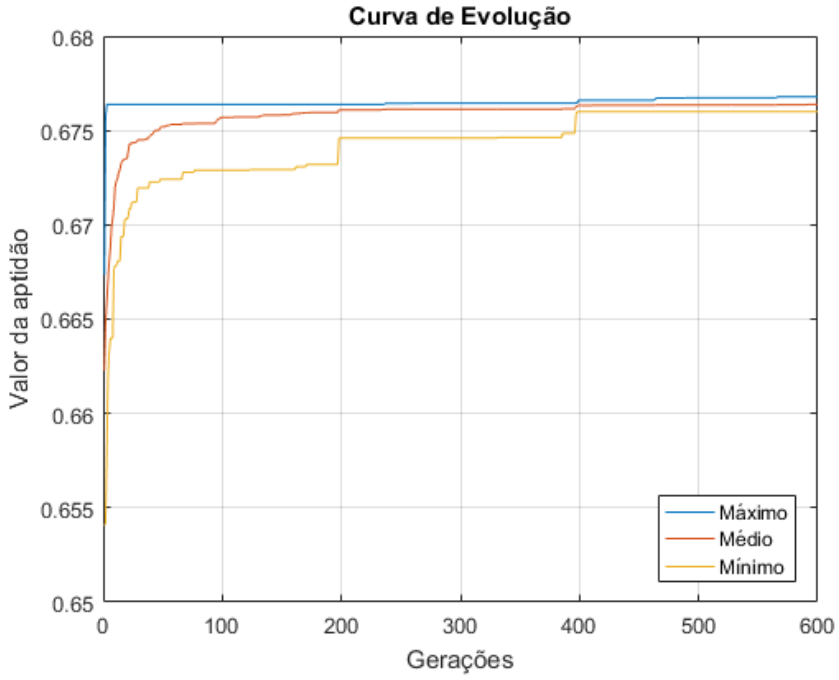

Figura 51 – Curva de evolução da base de dados Difficult2D, classe 2