UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Avaliação de Técnicas de Aprendizagem de Máquina como Surrogate na Otimização com Muitos Objetivos

Dissertação de Mestrado

Joel Alves de Oliveira

São Cristóvão – Sergipe

2020

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Joel Alves de Oliveira

Avaliação de Técnicas de Aprendizagem de Máquina como Surrogate na Otimização com Muitos Objetivos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Dr. André Brito Carvalho

São Cristóvão – Sergipe

2020

Este trabalho é dedicado a minha noiva, Paula Valéria, meu cunhado, Reneilson, meus pais, Marivaldo e Neuza Alves, meu irmão Josiel Fernando e meu orientador André Britto

Agradecimentos

Agradeço ao Prof. Dr. André Britto, pela forma exemplar em que conduziu a orientação das minhas atividades do mestrado. Sempre se mostrando solicito e presente para satisfazer as dúvidas que surgiram ao decorrer dessa jornada chamada mestrado.

Agradeço a Deus, por me conceder sabedoria e saúde para me dedicar durante esses dois anos de estudos. Foram muitos obstáculos e barreiras superadas. A rotina de trabalhar e estudar não é fácil, mais dos dias o cansaço aperta, então é preciso tirar forças de onde não existem e se sacrificar um pouco mais para que os objetivos sejam atingido ao final da jornada. Aqui deixo meus sinceros agradecimentos ao diretor de tecnologia da Atos Capital, Flavio Bento e ao gerente de tecnologia Deivid Marinho. Ambos foram coniventes e solicitos aos meus estudos e sempre apoiaram-me, seja flexibilizando os horários de trabalho, ou até mesmo liberando-me das atividades do trabalho para priorizar os estudos. Sou extremamente grato a todo o apoio dado pela equipe da Atos Capital nesse período em que cursei o mestrado.

Também agradeço de forma especial à Reneilson Yves, pelas caronas para ir as aulas do mestrado, e pelo conhecimento transmitido durante as disciplinas em que cursamos juntos. Por todas as correções e ajuda prestada na escrita dessa dissetação, assim como dos artidos que elaboramos e escrevemos juntos.

Agradeço a meus pais, Neuza Alves e Marivaldo Alves e meu irmão, Josiel Fernando. Sempre foram minha base e meu suporte, incentivando-me a seguir adiante e batalhar pelos meus sonhos e objetivos.

Por fim, agradeço a minha noiva e futura esposa Paula Valéria. Pessoa que não me deixou desanimar em momento algum, sempre esteve presente, dando-me força e estímulos para buscar o melhor, em meu trabalho, estudos, e na vida pessoal. Hoje sou um homem focado, dedicado e com objetivos bem definidos. Tenho a certeza que em tudo isso existe a mão dela por trás. Assim como idealizo que ao lado de um grande Homem existe uma grande Mulher.

Seu trabalho vai preencher uma parte grande da sua vida,
e a única maneira de ficar realmente satisfeito
é fazer o que você acredita ser um ótimo trabalho.
E a única maneira de fazer um excelente trabalho
é amar o que você faz.
(Steve Jobs)

Resumo

Em problemas de otimização existem um subconjunto de problemas que são definidos como problemas complexos, os quais apresentam modelagens de complexidade alta. Para essa classe de problemas existe um número exaustivo de possíveis combinações para as variáveis de entrada de um sistema. Assim, avaliar essas combinações é um processo humanamente inviável, então recorre-se a mecanismos de otimização que visam encontrar a melhor solução, dentre os quais é possível quantificar o grau de adequação das soluções às necessidades em causa. Geralmente, quando se tratam de problemas com até três funções objetivos são empregados Algoritmos Evolutivos para resolvê-los. Outra abordagem empregada é o uso de *surrogates*, os quais podem ser definidos como mecanismos capazes de aprender o comportamento de uma dada função. Ao usar esses mecanismos em problemas complexos estima-se obter como ganho a redução do alto custo computacional para computar os valores de *fitness* das funções objetivos. Dentre os mecanismos de *surrogate* comuns na literatura destacam-se as técnicas de regressão linear e aprendizagem de máquina. A aplicação de *surrogates* em problemas com mais de uma função objetivo, problemas multiobjetivo, requer o uso de um modelo de aprendizagem para cada função, entretanto, recentes estudos têm obtido êxito em empregar um único surrogate para problemas com mais de uma função objetivo. Porém o uso de *surrogate* em problemas de otimização com mais de três funções objetivos ainda é uma área pouco explorada. Diante disso, esse trabalho tem como objetivo propor e avaliar novas abordagens de treinamento de *surrogate* associados a Algoritmos Evolutivos. Foram desenvolvidos dois *frameworks*, um aplicado a classe de problemas mono-objetivo e outro voltado para problemas de otimização com muitos objetivos. Os *frameworks* propostos tem como característica o emprego de diferentes abordagens de treinamento de *surrogate* e também diferentes maneiras de uso de técnicas de aprendizagem de máquina. Os *frameworks* foram submetidos a experimentos usando problemas *benchmark*, onde cada configuração dos algoritmos foi executada por vinte vezes e armazendas as métricas de desempenho. Para confirmar ou refutar as hipóteses, foi aplicado o teste estatístico de *Wilcoxon*. Os resultados evidenciam que as técnicas de aprendizagens de máquinas, Arvore de Decisão e *Random Forest* quando aplicadas como *surrogate* proporcionam resultados satisfatórios, além disso, as metologias de treinamento de *surrogate* aqui propostas, associadas aos algoritmos NSGA-II e SMPSO obtiveram resultados melhores ou iguais que os algoritmos do estado da arte (NSGA-II e MOEADD), na maioria dos experimentos realizados.

Palavras-chave: *surrogate*, aprendizagem de máquina, algoritmos evolutivos, problemas de otimização complexos.

Abstract

In optimization problems there is a subset of problems that are defined as complex problems, which present high complexity models. For this class of problems there is an exhaustive number of possible combinations for the input variables of a system. Thus, evaluating these combinations is a humanly unfeasible process, so we use optimization mechanisms that aim to find the best solution, among which it is possible to quantify the degree of adequacy of the solutions to the needs in question. Generally, when dealing with problems with up to three objective functions, Evolutionary Algorithms are used to solve them. Another approach employed is the use of surrogates, which can be defined as mechanisms capable of learning the behavior of a given function. When using these mechanisms in complex problems, it is estimated that the high computational cost reduction to obtain the fitness values of the objective functions will be gained. Among the common surrogate mechanisms in the literature, the techniques of linear regression and machine learning stand out. The application of surrogates in problems with more than one objective function, multiobjective problems, requires the use of a learning model for each function, however, recent studies have been successful in employing a single surrogate for problems with more than one objective function. However, the use of surrogate in optimization problems with more than three objective functions is still a little explored area. Therefore, this work aims to propose and evaluate new approaches to surrogate training associated with Evolutionary Algorithms. Two frameworks were developed, one applied to the class of mono-objective problems and the other aimed at optimization problems with many objectives.The proposed frameworks are characterized by the use of different approaches to surrogate training and also different ways of using machine learning techniques. The frameworks were subjected to experiments using benchmark problems, where each configuration of the algorithms was executed for twenty times and stores the performance metrics. To confirm or refute the hypotheses, the Wilcoxon statistical test was applied. The results show that the machine learning techniques, Decision Tree and Random Forest when applied as a surrogate provide satisfactory results, in addition, the surrogate training methodologies proposed here, associated with the NSGA-II and SMPSO algorithms obtained better or equal results. than state-of-the-art algorithms (NSGA-II and MOEADD), in most of the experiments carried out.

Keywords: Surrogate. machine learning. evolutive algorithms. complex optimization problems

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Lista de abreviaturas e siglas

ASF	*Achievement Scalarizing Function*
EA	*Evolutionary Algorithm*
AE	Algoritmo Evolucionário
IGD	*Inverted Generational Distance*
KKT	*Karush-Kuhn-Tucker*
KKTPM	*KKT Proximity Measure*
MaOP	*Many-Objective Optimization Problem*
MOP	*Multi-objective Optimization Problem*
MSL	Mapeamento Sistemático da Literatura
RBF	*Radial Basis Functions*
RNA	Redes Neuronais Artificiais
RMHC	*Random Mutation Hill Climber*
rGA	*real-parameter Genetic Algorithm*
SLM	*Systematic Literature Mapping*
SVM	*Support Vector Machine*

Sumário

1 Introdução

Otimização pode ser definida como o estudo de problemas em que se procura minimizar ou maximizar uma função através de uma busca. Durante o processo de busca, procura-se uma solução com valores aceitáveis, de acordo com o tomador de decisão (função objetivo). Nesse contexto existem diversos problemas das áreas de Engenharias, Física, Química e outras que precisam usar técnicas de otimização para identificar boas soluções (COELLO, 2017). O processo de otimização para problemas de muitos objetivos envolve a modelagem do problema, escolha das variáveis de decisão que controlam as características do sistema e definição das funções objetivo (medidas quantitativas da qualidade do processo). Em geral, cada função objetivo retorna um valor real, assim, um conjunto de funções objetivos é representado por um vetor de número reais. As funções objetivo devem ser maximizadas ou minimizadas.

Quando um problema possui mais de uma função objetivo ele é definido como um Problema de Otimização Multiobjetivo (MOP, do inglês *Multi-objective Optimization Problem*). Tais problemas podem ser descritos ainda como um vetor de variáveis de decisão, satisfazendo as restrições dadas. Quando problemas multiobjetivos apresentam quatro ou mais objetivos eles são designados como Problemas de Otimização com muitos Objetivos (MaOP, do inglês *Many-Objective Optimization Problem*) (ISHIBUCHI; TSUKAMOTO; NOJIMA, 2008).

Os problemas de otimização requerem que o algoritmo avalie uma vasta quantidade de entradas para encontrar o conjunto das soluções ótimas de um dado problema. Quando os algoritmos estão trabalhando com problemas do mundo real são encontradas algumas limitações (COELLO, 2017). A principal limitação que pode ser destacada é o tempo gasto para o cálculo dos valores objetivos das funções que compõem essa classe de problemas.

A classe de problemas do mundo real é classificada como problemas *Expensive Optimization Problems*. Essa classificação se dá principalmente pela complexidade em computar os valores objetivos. Tal complexidade faz com que, na maioria das vezes, sejam empregados algoritmos meta-heurísticos para solucionar problemas reais. No decorrer da dissertação, a classe

de algoritmos meta-heurísticos será abordada como algortimos evolutivos [1]. Tomando como exemplo o problema de sincronização de semáforos, o qual é definido como uma abordagem que trata o problema de redução de congestionamento de tráfego (MATOS, 2017). Para esse problema não existe uma formulação matemática para calcular os valores objetivos, o que torna o problema complexo, uma vez que é necessário um simulador para calcular as funções objetivo.

Esses aspectos têm motivado os pesquisadores a buscarem novas alternativas para melhorar o desempenho, em termos de tempo de execução e qualidade das soluções geradas, dos algoritmos evolutivos (AE), quando aplicados a problemas complexos. Algumas abordagens empregadas visando diminuir o tempo de execução para os algoritmos evolutivos que se destacam são: o uso de abordagens de programação paralela, o uso de *surrogate* e semelhança dos vizinhos (COELLO, 2017). Quando refere-se a *surrogate*, define-se que qualquer método que possui a capacidade de aprender o comportamento de uma determinada função pode ser usado para substituir essa função durante a execução do AE. Como exemplos, têm-se os métodos de aprendizagem de máquina, regressão linear e regressão gaussiana. *Surrogates* partem do princípio de substituir uma ou mais funções objetivo por um mecanismo de predição. Para problemas complexos, esse mecanimo é mais barato computacionalmente que a função objetivo substituída. Assim, através da aplicação de *surrogates* serão executados um menor número de chamadas às funções objetivo (as funções objetivos serão substituidas pelos modelos treinados durante a busca do AE) e, consequentemente, ocorrerá uma redução no tempo de execução.

Em (COELLO, 2017; JIN, 2011) são apresentadas explanações, em forma de um *survey*, que permitem construir uma ideia de como se encontra essa linha de pesquisa nos últimos anos. Nesses estudos são abordadas as técnicas *neural networks* (NN), *radial-basis function* (RBF), *support vector machines* (SVM) e *Kringing*, como as principais metodologias de *surrogate* aplicadas atualmente. Isso evidencia que a área ainda está no início, existindo, portanto, espaço para exploração de uma gama de novas abordagens de aprendizado aplicadas em forma de *surrogate*.

Aproveitando esse fator, esse trabalho visa a criação de dois novos *framewoks*, em que, o primeiro *framework* foi desenvolvido e empregado em problemas mono-objetivos. O primeiro *framework* foi criado para validar as novas abordagens de treinamentos de *surrogate* propostas. O primeiro *framework* foi expandido e aplicado em problemas com muitos objetivos, criando-se assim o segundo *framework*. juanto ao segundo *framework* foram implementadas novas metodologias de treinamento de *surrogates*. Esse trabalho também irá explorar técnicas de aprendizagem de máquina ainda não abordadas como *surrogate*.

[1] Nem todas as meta-heurísticas aplicadas na área de otimização são do paradigma da computação evolucionária, porém, o termo algoritmos evolutivos é comumente utilizado na literatura para definir toda a classe de algoritmos meta-heurísticos

1.1 Motivação

Em virtude do elevado tempo de execução dos algoritmos evolutivos quando aplicados a problemas complexos, se faz necessário encontrar novos métodos que possam ajudar a reduzir o número de chamadas às funções objetivo dessa classe de algoritmo, principalmente quando aplicado a problemas com muitos objetivos. É necessário buscar novas metodologias para serem aplicadas em algoritmos evolutivos visando a solução de problemas com muitos objetivos.

O estudo (ROY; HUSSEIN; DEB, 2017) propõem novas formas de treinamentos de *surrogate* associados a AE. A nova metodologia é classificada de M1 a M6. Nessa metodologia, as abordagens M5 e M6 propõem usar apenas um modelo para aprender todas as funções objetivos presente no problema de otimização. Isso é algo inovador, com resultados estatisticamente melhores ou iguais a AE clássicos da literatura, porém ainda pouco explorado. Outro aspecto é que os trabalhos empregaram, na grande maioria, as mesmas técnicas de aprendizagens. Assim, existe espaço para o aprimoramento e implementação de novas abordagens de alimentação de *surrogate*, assim como o uso de técnicas de aprendizagem de máquina ainda não testadas como *surrogate*.

Frente ao exposto, estima-se a possibilidade de contribuir para os algoritmos assistidos por *surrogate* no sentido de fazer uma análise experimental e exploratória dos seguintes quesitos:

- Abordagens de Aprendizagem de Máquina usadas como *surrogate*;
- Implementação de novas metodologias de alimentação de *surrogate* aplicado em problemas com muitos objetivos;

1.2 Problemática

Problema

Problemas de otimização complexos dispõem de um alto custo computacional para avaliação de uma dada entrada, além de que, na maioria dos casos, esses problemas possuem muitas funções objetivos, ou seja, englobam a classe de problemas de otimização com muitos objetivos. Solucionar esse tipo de problema requer a avaliação de um número exaustivo de soluções, o que leva muito tempo e deixa o algoritmo de otimização custoso. Assim, se faz necessário buscar novas abordagens que possam ser associadas à execução do algoritmo evolutivo, melhorando o desempenho deste para essa classe de problemas.

1.3 Objetivos

O objetivo deste trabalho é propor e avaliar novas abordagens de treinamento de *surrogate* associados a AE. As novas abordagens de treinamento consistem da metodologia empregada para

disponibilização dos dados ao modelo de aprendizagem aplicado como *surogate*. Serão avaliados as abordagens aqui propostas, assim como metodologias já disponíveis na literatura. Além disso, será avaliado o desempenho de diferentes modelos de aprendizagem na forma de *surrogate*.

Objetivos Específicos

São objetivos específicos:

- Propor abordagens de treinamento de *surrogates* associadas a técnicas de aprendizagem de máquina.
- Sugerir o melhor modelo de aprendizagem de máquina para ser usado como *surrogate* em AE, levando em consideração as classes de problemas usados e o contexto de problemas com muitos objetivos;
- Avaliar diferentes técnicas de *surrogate* em AE para problemas mono-objetivos e com muitos objetivos;

1.4 Justificativa e Contribuições

Justificativa

Embasado no conceito que a produção científica visa apropriar-se da realidade para melhor analisá-la e, posteriormente, produzir melhores soluções para os problemas encontrados, a discussão sobre os impactos do uso de *surrogate* em algoritmos evolutivos possui relevância no âmbito acadêmico.

Nesse contexto, o uso de diferentes metodologias e abordagens de *surrogates* atribuídas a algoritmos evolutivos pode ser o início de um processo que começa na academia e estende seus reflexos para a realidade cotidiana.

Contribuições

O trabalho apresenta como principais contribuições um mapeamento do estado da arte, que elenca as principais metodologias usadas como *surrogate* em AE. Além disso, serão testadas modelos de aprendizagem de máquina ainda não usados como *surrogate*. Ao final do trabalho são apresentados dois novos *frameworks*, nos quais são empregadas novas abordagens de alimentação de *surrogates* e diferentes maneiras de uso de técnicas de AM.

1.5 Metodologia

O projeto encontra-se dividido em duas etapas, sendo a primeira etapa o uso de *surrogates* aplicados a problemas de otimização mono-objetivos e a segunda etapa o uso de *surrogate* aplicado a problemas de otimização com muitos objetivos.

Ambos as etapas possuem quatro fases distintas, sendo a primeira fase a realização de uma revisão bibliográfica sobre algoritmos evolutivos assistidos por *surrogate*. Assim, ter-se-á uma compreensão sobre como está o estado da arte (produção de artigos, dissertações, teses) nesta área de pesquisa. Com isto, pretende-se ter um embasamento teórico suficiente para o desenvolvimento do projeto.

Conseguinte ao estudo bibliográfico, serão comparados os algoritmos de aprendizagem de máquina obtidos a partir da revisão bibliográfica feita, de forma a descobrir qual o melhor algoritmo de aprendizagem de máquina para ser empregado como *surrogate* em AE.

Na terceira parte (realizada em paralelo com a segunda), serão selecionados e implementados novos algoritmos de aprendizagem de máquina ainda não aplicados na literatura como *surrogates*. Nesse contexto foram escolhidos os algoritmos de Árvore de Decisão e *Random Forest*, por serem algoritmos de apresendizagem de máquina com bom desenpenho como regressor e nunca terem sido usados na forma de *surrogate* , assim como o algoritmo SVM (já implementados em alguns trabalhosna literatura).

Nessa terceira fase, em paralelo à seleção de novos algoritmos de aprendizagem de máquina, serão propostos dois novos *frameworks*, os quais englobam o uso de novas abordagens de alimentação e treinamento do *surrogate* e diferentes formas de uso de técnicas de aprendizagem de máquina.

Na quarta parte, tendo em vista a conclusão da segunda e terceira, serão realizados testes de desempenho dos algoritmos através de simulações, obtendo dados de eficácia. Cada configuração de algoritmo formada por abordagem de treino do *surrogate* - técnica de aprendizagem de máquina - algoritmo evolutivo, será executada vinte ou mais vezes, onde em cada execução será armazenado o melhor valor objetivo de cada função objetivo, ao final será tirada a média e desvio padrão das vinte medidas.

Ainda na quarta etapa, os resultados coletados serão submetidos a testes estatísticos, uma forma de mitigar possíveis erros de avaliações, além de dar uma maior credibilidade às análises dos resultados.

Ressalta-se que o primeiro estudo sobe o uso de *surrogate* aplicado a problemas mono-objetivos possui um caráter investigatório e tem como finalidade servir de base para o estudo principal da dissertação, estudo sobre o uso de *surrogate* aplicado a problemas com muitos objetivos.

1.6 Estrutura do Trabalho

Para melhor compreensão da proposta, o texto está organizado nos seguintes capítulos:

- No capítulo 2 são abordados refenciais teóricos sobre otimização e *surrogate*, problemas

com muitos objetivos e as técnicas de aprendizagem de máquina usadas, assim como os principais trabalhos da literatura para esse tema.

- No capítulo 3 são apresentadas a metodologia seguida nessa proposta, assim como as abordagens *surrogate* propostas e implementadas e os resultados iniciais obtidos. E por fim os desafios futuros e o cronograma a ser seguido.
- O último capítulo, capítulo 5, apresenta as considerações finais a respeito do que já foi desenvolvido até o momento.

2

Fundamentação Teórica

Este capítulo aborda conceitos básicos sobre otimização multiobjetivos e com muitos objetivos, assim como traz uma definição para *surrogate*. Aqui também são apresentadas explanações a respeito das técnicas empregadas na forma de *surrogate* em AE. Por fim é realizada uma breve explanação dos principais artigos que usam *surrogate* em Algoritmos Evolutivos.

2.1 Otimização Multiobjetivo

Otimização consiste em uma metodologia capaz de obter as melhores configurações aceitáveis para o funcionamento de um sistema. Um dado problema de otimização tem sua solução a partir de um processo de exploração do espaço de busca. Nesse processo soluções são construídas e avaliadas, onde a avaliação é feita utilizando uma função objetivo. A partir dessa avaliação é possível comparar duas soluções e verificar qual delas é melhor (COELLO et al., 2007a).

Os MOPs possuem mais de uma função objetivo e cada instância de um MOP pode ser representada por uma dupla (Ω, f). Onde Ω é o conjunto das soluções candidatas e f é a conjunto de funções objetivos responsável por avaliar cada solução $x \in \Omega$. Nesse contexto espera-se encontrar uma solução $x^* \in \Omega$ que minimize ou maximize o conjunto de funções objetivos $f(x)$.

Segundo (COELLO et al., 2007a), um problema multiobjetivo sem restrições é definido como:

$$f(x) = (f_1(x), ..., f_n(x)), x \in \Omega$$

Dada uma solução x de MOP, ela irá minimizar (ou maximizar) os componentes de um vetor $f(x)$ onde x é um vetor composto por uma variável de decisão n-dimensional, $x = (x_1, ..., x_n)$, a partir do conjunto de soluções Ω. O conjunto de soluções Ω corresponde a todas as possíveis soluções x que podem ser utilizadas para satisfazer uma avaliação de $f(x)$.

Em problemas MOP é preciso encontrar uma solução que otimiza as m funções objetivos. Se faz necessário otimizar as m funções objetivos de forma simultânea, dessa forma não vai existir mais apenas uma solução x e sim um conjunto das melhores soluções x, as quais atendem aos critérios da Teoria de Otimalidade de Pareto (COELLO et al., 2007a). Para problemas MOP o ótimo é encontrado a partir dos conceitos a seguir:

- Dominância de pareto: dado duas soluções de entrada x e y representadas pelos seus vetores de variáveis no espaço objetivo, $f(x) = (f_1(x), ..., f_n(x))$ e $f(y) = (f_1(y), ..., f_n(y))$, respectivamente, podemos dizer que $f(x)$ domina $f(y)$ (representado por $f(x) \preceq f(y)$) se, e somente se, $f(x) \preceq f(y)$ em todas as dimensões, ou é melhor em uma dimensão. Assim temos a seguinte representação $\forall_i \in \{1, ..., m\}, f(x)_i \leq f(y)_i \wedge \exists_i \in \{1, ..., m\} : f(x)_i < f(y)_i$

- Otimalidade de pareto: uma solução x é dita ótimo de pareto se não existir nenhum vetor $f(y)$ que consiga melhorar algum objetivo sem causar prejuízo simultâneo em pelo menos um outro objetivo. A solução $f(x)$, é classificada ótimo de pareto se o seu vetor de funções objetivo é não dominado. Por definição uma solução que é ótimo de pareto possui um vetor de funções objetivos que não pode ser simultaneamente melhorado (CARVALHO, 2013).

- Conjunto Ótimo de Pareto: em um MOP, o Conjunto Ótimo Pareto, P^* , é o conjunto das melhores soluções pertencentes a Ω, em outras palavras, é o conjunto de soluções do problema que são ótimo pareto (CARVALHO, 2013).

- Fronteira de Pareto: cada solução pertencente a P^* possui uma imagem de um ponto não dominado. O conjunto desses pontos não dominados no espaço das funções objetivos são denominados de fronteira de pareto (CARVALHO, 2013).

2.1.1 Problemas com muitos Objetivos

A grande maioria dos algoritmos MOEA funcionam bem em classe de problemas com apenas duas funções objetivos. Porém, ao aumentar o número de funções objetivo, esses algoritmos têm a sua performance de busca prejudicada (CARVALHO, 2013).

A classe de problemas multiobjetivo com quatro ou mais objetivos são designadas de Problemas de Otimização com Muitos Objetivos (ISHIBUCHI; TSUKAMOTO; NOJIMA, 2008). Quando o número de funções objetivo do problema cresce, os algoritmos evolucionários multiobjetivos que realizam a busca baseada na dominância de Pareto apresentam algumas dificuldades para encontrar as melhores soluções, ou seja, as soluções ótimas. Visto que existem problemas do mundo real que tendem apresentar uma grande variedade de funções objetivos se faz necessário desenvolver MOEAs para lidar com Problemas de Otimização com muitos Objetivos.

Essa classe de problemas traz consigo uma série de desafios que precisam ser tratados por qualquer algoritmo de otimização, incluindo os MOEAs. Em primeiro lugar, a proporção de soluções não dominadas tende a crescer exponencialmente, uma vez que essas soluções são representadas por conjuntos de vetores objetivos, e a quantidade de objetivos têm aumentado. Além disso existem alguns outros desafios a serem sanadaos como, maioria das soluções são não-dominadas, Avaliação de métrica custos e processo de recombinação ineficiente(DEB; JAIN, 2014; CARVALHO, 2013).

Uma abordagem que tem sido emprega em problemas MOP, porém ainda não testada em problemas MaOP são os *surrogates*. Essa abordagem tem obtido resultados próximos aos algoritmos do estado da arte quando aplicada em problemas com até três funções objetivos (ROY; HUSSEIN; DEB, 2017), e pode ser um caminho promissor a ser explorado junto aos algoritmos evolutivos quando aplicado a problemas MaOP. Outra característica que torna os *surrogates* uma boa opção é a capacidade de aprender o comportamento das funções objetivos, geralmente complexas, e então passar a substituir as funções objetivas durante a busca. Com isso é esperado que o tempo de execução do AE diminua, uma vez que levar-se-a menos tempo para avaliar as soluções a partir do surrogate.

2.2 *Surrogates*

Entende-se por *surrogate* modelos computacionais eficientes usados para aproximar a avaliação custosa de um problema de otimização e, com isso, diminuir o número total de avaliações.

O autor entende *surrogate* como uma técncica que irá aprender a função objetivo ($f(x)$) e, após isso, será usada para predizer os valores de $f(x)$ dada uma solução x. Existem inúmeras abordagens que podem ser usadas como *surrogate*, algoritmos de aprendizagem de máquina, algoritmos de regressão linear, dentre outros (COELLO et al., 2007a).

Essas abordagens que podem ser empregadas como *surrogate* precisam de treinamento. No processo de treinamento são diponibilizados as soluções Ω assim como os valores de *fitness* $f(x) = (f(x_1)...f(x_n))$ associados a cada solução, respectivamente. Com esses valores as abordagens buscam construir um padrão que relacionam as entradas aos seus respectivos valores de saída.

Uma vez treinado, o *surrogate* será empregado no lugar da função objetivo, ou seja, dada como entrada uma solução x, o *surrogate* retornará um valor $f(x)$. Vale salientar que o *surrogate* corresponde a uma aproximação da função objetivo, então, o valor predito por este não corresponde ao valor real, que seria retornado pela função objetivo real.

Nas subseções seguintes são discutidas as taxonomias de *surrogate* encontradas na literatura. Após essa discussão as seções seguintes apresentam as definições para as técnicas de

aprendizagem de máquina usadas como *surrogate* em AE nessa dissertação.

2.3 Modelos de aprendizagem

Esta seção apresenta as principais técnicas de aprendizagem de máquina usadas na literatura como *surrogate* em problemas de otimização.

2.3.1 Máquina de Vetores de Suporte para Regressão

Originalmente desenvolvidas para resolver problemas de reconhecimento de padrões, o método SVM foi posteriormente estendido para o problema de regressão, ou seja, aproximação de funções (BRANKE et al., 2008). Quando expandido para problemas de regressão as características do método SVM persistiram, e foi implementada a função de perda ϵ-insensível.

A introdução da abordagem de vetores de suporte ao problema de regressão é feita através da determinação de um tubo de raio ϵ, o qual deverá conter todos os pontos do conjunto de treinamento. Esse valor representa a máxima perda admitida para cada ponto do conjunto.

Assim, apresentado um conjunto de treinamento Z, a função $f(x) = wx_i + b$, é obtida através da minimização do risco a seguir:

$$R_{reg} = \frac{1}{2} \|W\|^2 + Cl_\epsilon(y, x, f) \tag{2.1}$$

onde $Cl_\epsilon(y, x, f)$ dimensiona o risco associado à função de perda ϵ-insensível. A minimização da norma quadrática $\|W\|^2$ é aplicada buscando reduzir a complexidade da função estimada e também buscando uma maior generalização do método.

Como forma de assegurar a viabilidade do problema introduz-se, para um dado raio ϵ, um conjunto de restrições associadas às variáveis de folga, ξ_i e ξ_i^* , para cada ponto pertencente ao conjunto de dados.

$$\begin{gathered} y_i - wx_1 - b \leq \epsilon + \xi_i, \quad para\ wx_1 + b \leq y_i \\ wx_1 + b - y \leq \epsilon + \xi_i^*, \quad para\ wx_1 + b \geq y_i \\ \xi_i, \xi_i^* \geq 0 \end{gathered}$$

A partir da formulação das restrições é possível chegar ao problema de otimização quadrática restrito para regressão SVM do tipo primal (BRANKE et al., 2008), como apresentado

a seguir:

$$\text{minimizar} : \frac{1}{2} \|W\|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \tag{2.2}$$

$$\text{Sujeito a} : \begin{cases} y_i - wx_1 - b \leq \epsilon + \xi_i \\ wx_1 + b - y \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{2.3}$$

Onde a constante $C > 0$ irá ajustar o compromisso entre a complexidade do modelo e a quantidade de desvios maiores que ϵ.

2.3.2 Árvore de Decisão para Regressão

O uso de árvore binária para regressão foi usado incialmente por Morgan e Sonquist, aplicadas no programa AID (*Automatic Interaction Detection*). Posteriormente essas técnicas foram estendidas, originando o algoritmo CART (*Classification and Regression Tree*) (ARGETON, 2013).

2.3.2.1 Treinamento

O treinamento realizado por esse algoritmo tem por característica particionar o domínio recursivamente, de forma binária, ou seja, cada nó da árvore pode tomar apenas dois caminhos, essa característica tem como finalidade aumentar a homogeneidade de cada nó.

Ao finalizar o processo de partição, os nós terminais da árvore (nó de termino) estão associados a uma classe de problema, para os problemas de classificação ou a um valor real quando aplicado a problemas de regressão, já os demais nós possuem os atributos, que correspondem às variáveis de entrada do problema 1.

As principais ações realizadas para Treinamento da árvore de decisão podem ser elencados como segue:

- Determinar todas as possíveis partições que um dado nó pode realizar, ou seja, quais caminhos seguir. Geralmente é usado particionamento binário;
- Selecionar o melhor caminho, melhor solução;
- Decidir quando um dado nó é terminal, denominado como nó folha;
- Atribuir um valor para cada nó folha.

Finalizado o procedimento de construção da árvore, é realizado um procedimento de poda. Neste procedimento é aplicado uma métrica que avalia a importância dos nós folhas para a árvore, os nós classificados como irrelevantes são descartados.

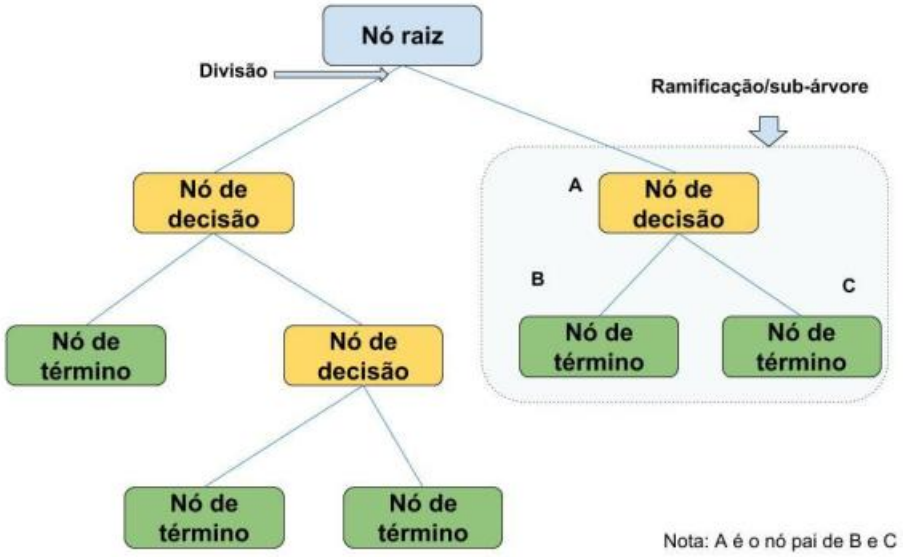

Figura 1 – Ilustração dos nós de uma árvore de decisão

O processo de treinamento da árvore está ligado a entropia, uma medida que mede a pureza de um conjunto (MINING, 2006). Dado um conjunto S, com instâncias pertencentes à classe i, com probabilidade p_i, tem-se:

$$Entropia(S) = \sum (P_i * \log_2 * P_i) \tag{2.4}$$

Com base na entropia é possível calcular o ganho para cada nó, ou seja, para cada atributo. Para isso é usado a relação a seguir:

$$Ganho(S, A) = Entropia(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} * Entropia(S_v) \tag{2.5}$$

O $Ganho(S, A)$ é calculado ordenando a árvore a partir do nó A, buscando uma redução para a entropia. A cada iteração do algoritmo os ganhos são calculados para todos os nós, e o melhor nó, aquele que possuir o maior ganho, irá ser usado como o nó topo. Quando todos os nós apresentarem entropia zerada, presupõe-se que a árvore de decisão está treinada.

2.3.2.2 Predição

Uma vez que a árvore está treinada a solução de entrada será submetida aos nós da rede, assim, cada nó irá fazer uma verificação e determinar o caminho a ser percorrido até chegar ao nó terminal da árvore, determinando assim o valor *fitness* que está associado a entrada. Cada nó da arvore possui um valor associado a ele, então, os valores de entrada serão comparados a esses valores construindo assim o caminho até o nó folha.

2.3.3 *Random Forest* para Regressão

A técnica de *Random Forest* parte do princípio de combinar árvores de decisão com agregação. Nelas (nas árvores), todas as amostras de agregação são distribuídas identicamente. Esse processo de distribuir as amostras de forma idêntica faz com que a média da esperaça das B árvores seja igual à esperança de cada árvore, isso implica que o observado no agregado de árvores consiste no observado em cada árvore.

Durante a treinamento, a cada iteração são selecionadas aleatoriamente as variáveis que serão utilizadas no treinamento das árvores de decisão. Ao usar D variáveis aleatórias identicamente distribuídas com variância σ^2 e correlação ρ para cada par, a variância média vai ser dada por:

$$\rho\sigma^2 + \frac{1-\rho}{D}\sigma^2 \tag{2.6}$$

Analisando a relação acima, observa-se que o segundo termo tende a zero quando o número de árvore aumenta, porém, o primeiro termo permanece intacto, portanto, para aumentar a variância, a floresta aleatória gera árvores mais idependentes entre si, diminuindo a correlação entre os classificadores.

Uma característica dessa técnica é a randomização das variáveis preditoras, isso torna necessário controlar o número de características (variáveis da entrada) que serão usadas em cada interação do algoritmo. Segundo (MINING, 2006), quando se tem $p^* << p$, é posível encontrar os melhores resultados. Sendo que p^* é um subconjunto de todas as variaveis de entradas (características dos dados) e p o conjunto total das variáveis de entrada.

A probabilidade de predição para a floresta aleátoria é calculada a partir da equação 2.7, dada as probabilidades para cada árvore, $\hat{p}_1, \hat{p}_2, ..., \hat{p}_B$.

$$\hat{p} = \frac{1}{B}\sum_{j=1} B\hat{p}_j \tag{2.7}$$

Ao final ocorre um processo de votação para identificar a árvore que obteve os melhores resultados. Assim, é mensurada a média do número de votos, que representa o quanto a regressão correta excede a média do número de votos para qualquer outra regressão, e, com isso, define-se qual árvore teve o melhor resultado.

2.4 Trabalhos Relacionados

Os estudos aqui apresentados foram mapeados a partir de um Mapeamento Sistemático da Literatura (MSL). O MSL teve como objetivo revisar as abordagens de *surrogate* empregadas em algoritmos de busca e otimização. Atráves deste, foi encontrado um total de duzentos e cinquenta e nove (259) artigos primários, sendo que ao final foram selecionados vinte e sete (27) artigos.

Para a seleção dos artigos do MSL foram adotados critérios de inclusão e exclusão para possibilitar a identificação impessoal e padronizada dos estudos relevantes

2.4.0.1 Critérios de Seleção MSL

Como critérios de inclusão (CI) foram adotados:

- CI1 - Incluir artigos que apresentam técnicas de surrogate aplicada a algoritmos evolutivos, no título, resumo ou palavras-chaves.
- CI2 - Incluir os artigos cujo método apresente ao menos uma técnica de surrogate aplicada a AE além de possibilitar a reprodução da abordagem usada e explicitar os problemas aos quais a abordagem foi aplicada.

Como Critérios de Exclusão (EC) foram definidos:

- EC1 - Excluir artigos duplicados.
- EC2 - Desconsiderar artigos que não disponibilizam o texto na íntegra.
- EC3 - Descartar trabalhos que usam técnica de surrogate porém não aplicadas ao contexto de AE.
- EC4 - Excluir qualquer trabalho que não esteja englobado no contexto do uso de surrogates aplicado a AE.

Ao referir-se às técnicas de surrogates definiu-se que seria buscado: (i) Kringing; (ii) Suport Vector Machine; (iii) Redes Neuronais Artificiais; (iv) Radial Basis Functions; (v) Gaussian Processes; (vi) Polynomial Regression; (vii) k-nearest neighbors; (viii) Lógica Fuzzy;

2.4.0.2 Etapas do processo de seleção

O processo de seleção foi baseado em duas etapas, aplicados em ambos os mapeamentos aqui realizados.

Na primeira etapa foram lidos os títulos, resumos e palavras-chaves dos artigos envolvidos no MSL. Durante essa etapa foram selecionados trinta e sete artigos. Na segunda etapa foi feita a leitura por completo artigos. Após esse processo, novamente foi aplicado os filtros de seleção, restando um total de vinte e sete artigos, ver tabela 1, identificados como relevantes, selecionados mediante os filtros de buscas e critérios de seleção apresentados.

Depois da aplicação dos critérios de seleção nota-se que a base de dados *ACM Digital Library* não possui nenhum trabalho entre os trabalhos relevantes selecionados, isso ocorreu perante a não disponibilização das informações do texto na íntegra por parte dessa base de dados.

Tabela 1 – Estudos relevantes selecionados (MSL)

Base de Dados	Quantidade de Artigos
Scopus	18
IEEE Xplore	1
ScienceDirect	8

Com a seleção finalizada, iniciou-se a extração dos dados para responder às questões de pesquisa. Também foi possível observar que durante o processo de seleção dos trabalhos, em cada uma das fases, os autores mantiveram-se em constante contato para a retirada de possíveis dúvidas ou conflitos, bem como tornar o artigo o mais imparcial possível.

Um aspecto detectado no MSL foi a falta de um comparativo entre as diversas técnicas que podem ser empregadas como *surrogate*, assim como abordagens de *surrogates* aplicadas a problemas com muitos objetivos.

Os estudos mapeados evidenciam algumas técnicas de aprendizagem de máquina e de regressão linear como as principais abordagens usadas como *surrogate* em algoritmos evolutivos. Dentre as técnicas com destaque estão *Multilayer perceptron*- (MLP) , *radial-basis function* (RBF), *support vector machines* (SVM) e *Kringing* (COELLO, 2017).

Nas próximas seções são abordados alguns dos principais estudos que foram mapeados, sendo que as seções estão organizadas de acordo com as técnicas empregadas como *surrogate*.

2.4.1 *Suport Vector Machine - SVM*

Os trabalhos de (ROSALES-PÉREZ et al., 2015; LIAU et al., 2013; ROSALES-PÉREZ et al., 2013) apresentam o uso de SVM para auxiliar algoritmos evolutivos a encontrarem os melhores conjuntos de soluções. São apresentadas abordagens diferentes, sendo a principal diferença a metodologia de treinamento do SVM, cuja finalidade é aprender o comportamento da função objetivo.

Na abordagem desenvolvida por (ROSALES-PÉREZ et al., 2015; ROSALES-PÉREZ et al., 2013) são usados dois arquivos externos, *arquivo A* e *arquivo B*, onde no *arquivo A* são armazenadas as soluções não dominadas encontradas durante o processo da busca, e no *arquivo B* é armazenado o conjunto de soluções avaliadas, juntamente com os valores de avaliação (*fitness*). Os estudos de (ROSALES-PÉREZ et al., 2015; ROSALES-PÉREZ et al., 2013) usam o *arquivo B* para treinar o modelo do *surrogate* durante a execução do algoritmo evolutivo, sendo que, a cada iteração do algoritmo, os arquivos externos são atualizados, assim como o modelo *surrogate*. Esse procedimento é realizado até que o conjunto de soluções ótimas para o problema seja encontrado.

Os três estudos compararam seus resultados com outros algoritmos, SUMO, PSMS e

NSGA-II, respectivamente. Em geral os resultados mostraram-se satisfatórios e pode-se destacar como contribuições principais o fato do uso de *surrogate* (SVM) reduzir o número de avaliações da função objetivo, além do menor tempo de execução do AE.

No estudo de (LIAU et al., 2013) o treinamento do SVM é realizado de forma *online*, ou seja, a medida que as soluções são geradas pelo algoritmo AE o modelo SVM é treinado. Para o treinamento, são usados os valores reais de avaliação da função objetivo, sendo que as soluções mais antigas são descartadas, utilizando-se apenas as mais recentes. Para validar os resultados do método proposto não é feita nenhuma comparação com outros AE da literatura. É feita apenas uma comparação dos resultados obtidos pelo método proposto para diferentes configurações de mutação e *crossover* no algoritmo AE abordado *multiobjective evolutionary algorithm based on decomposition* (MOEA/D).

No estudo de (ROSALES-PÉREZ et al., 2013) é usada a classe de problemas *ZDT test suite*, enquanto em (LIAU et al., 2013) são empregados um maior conjunto de problemas, ZDT, UF e *WFG problem test suite*. Já o estudo de (ROSALES-PÉREZ et al., 2015) usa o *benchmark* IDA. Todas essas classes de problemas têm como característica apresentar um número de objetivos inferior a três. Em todos os trabahos foram treinados apenas um modelo de SVM.

Os trabalhos citados acima apresentaram alguns pontos positivos, como a capacidade de determinar o pareto ótimo avaliando um número menor de soluções, além de redução do tempo de execução do algoritmo AE ao usar *surrogate* SVM. Porém existem alguns pontos críticos a serem abordados, em nenhum dos trabalhos foi feito testes com outras técnicas de *surrogate* já existentes.

2.4.2 Redes Neurais Artificiais

Os trabalhos de (ZĂVOIANU et al., 2017; ROY; HUSSEIN; DEB, 2017; ASCIONE et al., 2017) apresentam algoritmos evolutivos assistidos por uma rede neuronal artificial. Neles são empregados abordagens semelhantes, no qual a rede neuronal tem o papel de aprender o comportamento da função objetivo e, posteriormente, predizer o valor da função objetivo durante a busca do algoritmo evolutivo.

Nas abordagens apresentadas são usados algoritmos e problemas distintos, sendo que (ZĂVOIANU et al., 2017) usa o algoritmo DECMO2 aplicado a um problema de otimizar paramêtros de processos. A ideia é aplicar um algoritmo para otimizar parâmetros de processos que contribuem para constituir os melhores indicadores de controle.

A partir de uma entrada, conjunto de parâmetros de configurações, foi gerado um indicador. Esse processo de gerar os indicadores é custoso, pois necessita de simulações, então é treinada uma rede neural para que, dado um conjunto de parâmetros, tal rede estime os indicadores.

O algoritmo evolutivo realiza uma busca dos possíveis parâmetros. A medida que essa busca é realizada a rede neuronal é treinada, após isso ela substitui a função objetivo,

possibilitando que um maior número de entradas seja explorado.

No estudo de (ASCIONE et al., 2017) é empregado o algoritmo Moga para a resolução de um problema do mundo real (referenciar contruções em categorias). Nesse contexto a rede é treinada para determinar qual o cenário da restauração de uma dada construção.

O cenário predito pela rede neuronal é considerado a saída da função objetivo, e é usado pelo algoritmo evolutivo para o processo de otimização. Para esse problema existem milhares de potenciais contruções que precisam ser restauradas. Destacando-se que o processo de avaliação para definir se uma contrução está precisando de restauração requer uma simulação (que demora aproximadamente um minuto).

Assim, a rede neuronal aprende a função objetivo, e passa a substitui-la, predizendo os cenários que serão usados pelo algoritmo evolutivo. Como a rede neuronal leva menor tempo para avaliar uma entrada (características das construções), isso possibilita que milhares de potenciais construções sejam analisadas pelo algoritmo evolutivo, fato que era inviável usando o simulador.

Em (ROY; HUSSEIN; DEB, 2017) é usado o algoritmo N-RGA aplicado a problemas *benchmark*, ZTD, DTLZ, C2DTLZ. Este trabalho testa uma nova metodologia para o uso de *surrogate* em algoritmos evolutivos, entitulada de M6, nessa metodologia propõe-se usar uma única técnica de *surrogate* para problemas com mais de uma função objetivo, a maioria dos trabalhos empregam um *surrogate* para cada função objetivo existente no problema. O algoritmo N-RGA é empregado juntamente com uma rede neural (MLP) e um modelo de regressão *Kringing* como *surrogates*. Essa abordagem é empregada em problemas com no máximo três funções objetivos e mostrou-se promissora nos experimentos, sendo que a configuração usando a rede neuronal obteve os melhores resultados para os problemas com maior número de funções objetivos (2 e 3).

Dentre os três estudos destaca-se a nova abordagem proposta em (ROY; HUSSEIN; DEB, 2017), que apresentou resultados promisores, mostrando ser um caminho para novas pesquisas. Em nenhum dos trabalhos é feito algum tipo de comparação com outros trabalhos que usaram ou empregaram abordagens *surrogate* em algoritmos evolutivos. Os estudos que empregaram problemas do mundo real, (ZĂVOIANU et al., 2017; ASCIONE et al., 2017), apenas validaram os benefícios em termos de eficácia (melhores indicadores) e eficiência (redução do tempo de execução), respectivamente.

2.4.3 *Kringing*

O modelo gaussiano *Kringing* foi o modelo mais usado nos trabalhos analisados no mapeamento sistemático. Todos os trabalhos, (WANG; PENG, 2017; VOLZ; RUDOLPH; NAUJOKS, 2017; HAMDAOUI et al., 2015; ZHANG et al., 2014), empregam o modelo *Kringing* com a finalidade de aprender o comportamento da função objetivo, para posteriormente aplicar o modelo treinado para predizer o valor retornado pela função objetivo, dada uma entrada.

Os trabalhos de (ZHANG et al., 2014) usam o modelo *Kringing* para modelar funções objetivos do mundo real, sendo que o modelo é usado para idealizar uma aproximação do problema de bombas centrífugas de sucção. Para esse estudo foram aplicados apenas um *surrogate*, o modelo *Kringing*. Sendo que esse *surrogate* foi empregado para dois diferentes algoritmos evolutivos NSGA-II e MOEA/D.

O modelo *Kringing* é treinado a medida que o algoritmo realiza a busca. Quando o *surrogate* está treinado ele é usado para predizer os valores da função objetivo. Como variáveis de entradas, são dadas possíveis configurações dos parâmetros para construir as bombas, e serão obtidos os objetivos, magnitude de velocidade e pressão.

Nos demais trabalhos (VOLZ; RUDOLPH; NAUJOKS, 2017; HAMDAOUI et al., 2015) são usadas duas abordagens distintas. Em (VOLZ; RUDOLPH; NAUJOKS, 2017) é proposta um novo AE que usa o modelo *Kringing* para susbtituir a função objetivo e em (HAMDAOUI et al., 2015) é empregado o algoritmo NSGA-II auxiliado pelo modelo *Kringing* para predizer os valores objetivos de cada solução. Ambos os trabalhos validaram suas propostas usando problemas *benchmark* da literatura, BBOB-BIOBJ e ZTD respectivamente.

Esses trabalhos aplicaram apenas uma abordagem de *surrogate*. Em nenhum momento foi realizado algum tipo de comparação com outros algoritmos que também empregam *surrogate*, assim como em (HAMDAOUI et al., 2015) é apenas verificado se o algoritmo NSGA-II assistido pelo *surrogate* consegue otimizar os problemas testados.

Nenhum desses trabalhos que empregam o modelo *Kringing* para substituir a função objetivo em AE fez comparações com algum outro trabalho que empregou essa mesma metodologia. Os trabalhos de (WANG; PENG, 2017; VOLZ; RUDOLPH; NAUJOKS, 2017) comparam seus resultados com outros AE do estado da arte, NSGA-II e SA-EMOA respectivamente, constatando que os modelos propostos obtiveram resultados melhores, avaliando um menor conjunto de soluções.

2.4.4 Novas Taxonomias

Na literatura são encontradas muitas propostas de taxonomias de *surrogate*. Aqui destacam-se em especial a taxonomia proposta por (ROY; HUSSEIN; DEB, 2017). Trata-se de seis diferentes *frameworks* para o treinamento do modelo de aprendizagem (M1-M6).

A Figura 2 ilustra a relação existente entre os seis *frameworks* propostos. Os *frameworks* trabalham de acordo com a quantidade de funções objetivos apresentadas pelo problema.

Os *frameworks* M1 e M2 trabalham com funções objetivos de forma individual, ou seja, para um problema com N funções objetivos será necessário o uso de N modelos de aprendizagem. Para esses dois *frameworks* as restrições são tratadas de forma individual em M1 e em conjunto em M2.

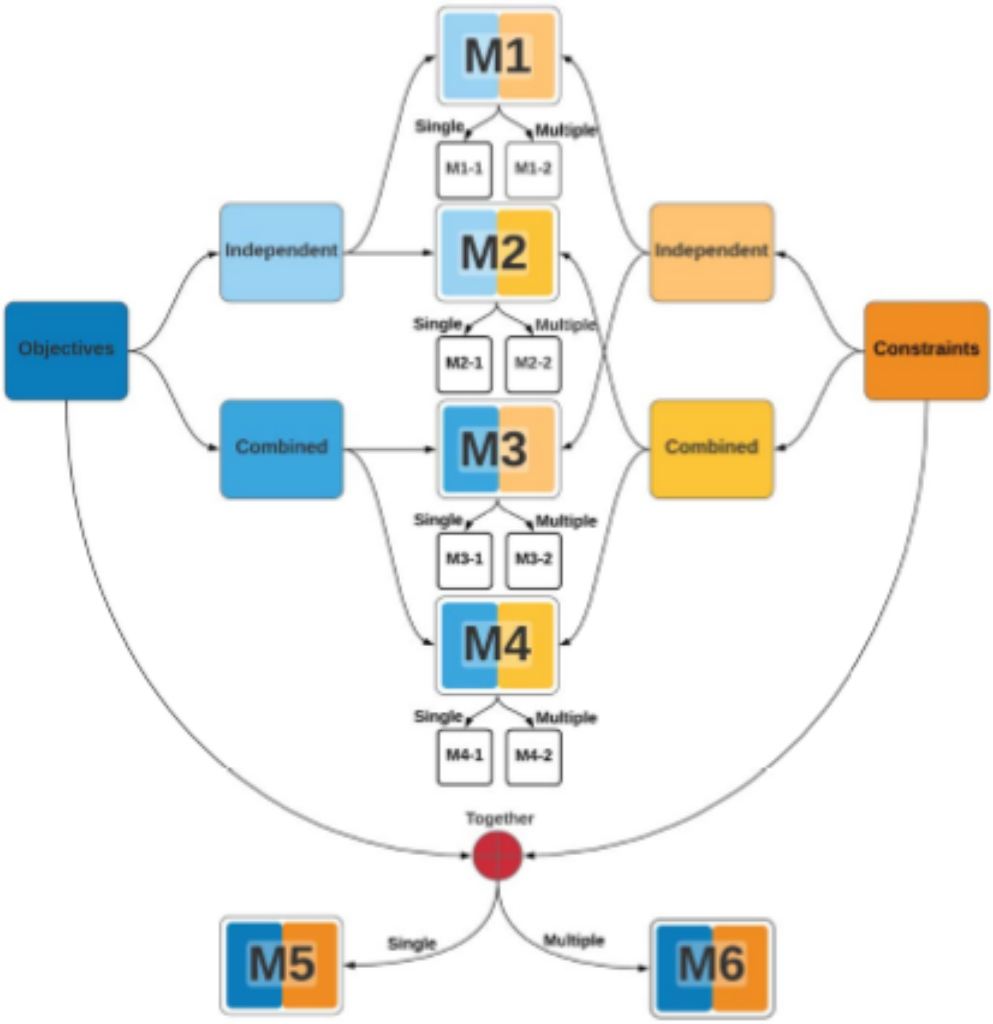

Figura 2 – Taxonomia proposta por (ROY; HUSSEIN; DEB, 2017) para seis diferentes *frameworks* de treino do modelo de aprendizagem

Nos *frameworks* M3 e M4 é feita uma transformação de um problema multiobjetivo para um problema de otimização simples. Esse processo é realizado usando *achievement scalarization function* - ASF.

Com isso, em M3 e M4, é construído apenas um único modelo de aprendizagem para aprender o problema simples de otimização gerado pela função ASF. Para as restrições dos problemas de otimização, esses dois *frameworks* trabalham de formas distintas, M3 emprega um modelo de aprendizagem para cada restrição do problema, enquanto M4 faz agregação de todas as restrições, construindo um único modelo de aprendizagem.

Diferente dos *frameworks* M3 e M4, o *framework* M5 usa a abordagem ASF para combinar todas as funções objetivos do problema assim como todas as funções de restrições.

Em M5 é construído um metamodelo do ASF resultante e este é usado como um problema

de otimização de objetivo único. Nesta abordagem é formulada uma função de seleção combinada $S(x)$ considerando todas as funções objetivos e todas as funções de restrição juntas, mas o foco final é otimizar globalmente a função S.

$$\text{S(x)} = \begin{cases} ASF(x), \text{ se x é ótima} \\ ASF_{max} + CV(x), \text{ caso contrário} \end{cases} \tag{2.8}$$

O último *framework*, M6, tem como principal característica o uso de um único modelo de aprendizagem para trabalhar com problemas de otimização multiobjetivos. Isso é possível graças ao emprego da métrica *KKT proximity measure* (KKTPM)(DEB; ABOUHAWWASH, 2016), que identifica o grau de dominação de uma solução. Esta métrica foi criada baseada nas condições de otimalidade de *Karush-Kuhn-Tucker* (KKT), e retorna como resultado valores dentro do intervalo [0-1].

Ao empregar a métrica KKTPM as soluções que pertencem ao conjunto de soluções ótimas apresentam valores próximos a zero, e soluções que não pertencem ao conjuntos de soluções ótimas apresentam valores mais elevados, próximos a um. Esses valores são usados para treinar o modelo de aprendizagem.

Outro trabalho recente que traz novas contribuições em termos de *surrogate* é o estudo de (PORTELLI; PALLEZ, 2019). Em seu trabalho, (PORTELLI; PALLEZ, 2019) propõe um novo algoritmo que usa diferentes *surrogates* para realiza a aproximação dos objetivos, aumentando assim a gama de funções que podem ser aproximadas. Também são usados dois conjuntos de vetores de referência (VR). As soluções são atíbuidas aos conjuntos de RF de forama separada e o conjunto que apresentar a melhor métrica de diversidade é escolhido como o vetor de referência vencedor.

No trabalho de (PORTELLI; PALLEZ, 2019) também é aplicado um esquema de busca local. Esse esquema tem como finalidade melhorar as soluções geradas. A busca local usa como base a métrica de distância euclideana. Por fim, para a geração de novas soluções, o algoritmo emprega as soluções pertencentes a um arquivo A. Nesse arquivo A são armazenadas apenas as soluções avaliadas como boas soluções, com isso, as novas soluções geradas tendem a serem boas também.

Um ponto relevante do trabalho de (PORTELLI; PALLEZ, 2019) está no comparativo feito com outros trabalhos que empregaram *surrogate*. O algoritmo proposto foi empregado em um conjunto de problemas *benchmark*, DTLZ, WFG, C1_DTLZ1, C2_DTLZ2, C3_DTLZ3 e comparado com algoritmos como ParEGO, MOEA/D-EGO, CSEA, K-RVEA. Observa-se resultados promissores nos experimentos, destacando-se que o algortitmo proposto obteve o conjunto de melhores resultados para a maioria dos exerimentos.

Para *surrogates* o artigo de (PORTELLI; PALLEZ, 2019) empregou as abordagens, *Kringing*, *polynomial response surface method* (RSM) e RBF. Todas essas abordagens já foram

empregadas na literatura em forma de *surrogate*, assim, seria interessante empregar novos métodos de aprendizados atrelados ao algoritmo proposto.

2.5 Resumo

Dentre os trabalhos relacionados foram aplicados cinco distintos modelos de aprendizagens como *surrogates*, sendo eles, *Kringing*, ANN e SVM, RBF e RSM. Destaca-se que todos esses modelos apresentados possuem a capacidade de atuar como um regressor linear.

Tabela 2 – Problemas e modelos aplicados nos trabalhos relacionados

Artigos	Modelo Aprendizagem	Tipo de Problema	Família do Problema
(LIAU et al., 2013)	SVM	multi-objetivo	ZDT
(ROSALES-PÉREZ et al., 2013)	SVM	multi-objetivo	ZDT, UF e WFG
(ZĂVOIANU et al., 2017)	MLP	multi-objetivo	*Optimize process Parameters* - Problema Real
(ROY; HUSSEIN; DEB, 2017)	MLP	multi-objetivo	ZTD, DTLZ, C2DTLZ
(WANG; PENG, 2017)	Kringing	multi-objetivo	*emergency libration point orbits transfer between sun-earth system and the earth-moon system* - Problema Real
(VOLZ; RUDOLPH; NAUJOKS, 2017)	Kringing	multi-objetivo	BBOB-BIOBJ
(HAMDAOUI et al., 2015)	Kringing	multi-objetivo	ZDT
(ZHANG et al., 2014)	Kringing	multi-objetivo	*Suction centrifugal pump problem* - Problema Real
(DEB et al., 2018)	ANN and Kringing	multi-objetivo	ZTD, DTLZ, C2DTLZ
(DEB et al., 2017)	ANN and Kringing	multi-objetivo	ZTD, DTLZ, C2DTLZ
(HUSSEIN; DEB, 2016)	ANN and Kringing	multi-objetivo	ZTD, DTLZ, C2DTLZ
(PORTELLI; PALLEZ, 2019)	Kringing, RBF and RSM	multi-objetivo	DTLZ, WFG, C1_DTLZ1, C2_DTLZ2, C3_DTLZ3

Os trabalhos empregaram o uso de *surrogate* com a finalidade de solucionar problemas de otimização considerados complexos, sendo alguns deles problemas *benchmark* da literatura e também problemas do mundo real. Na tabela 2 é apresentada uma planilha que evidência o modelo de aprendizagem, o tipo de problema e também a família de problemas empregado por cada trabalho.

3

Surrogate Aplicado a Problemas de Otimização Mono e Com Muitos Objetivos

A forma como os dados são fornecidos ao *surrogate* pode variar, possibilitando o emprego de diferentes metodologias de treinamento. Nessa dissertação propõe-se dois novos *frameworks*, os quais se baseam nos métodos de treinamento dos modelos de aprendizagem de máquina. Inicialmente foi implementado um *framework* no âmbito de problemas mono-objetivos. Nesse cenário foram feitos testes buscando ratificar que as metodologias de alimentação dos *surrogate* propostas são capazes de aprender o comportamento da função objetivo. Posteriormente as abordagens de alimentação de *surrogate* empregadas nas classes de problemas mono-objetivos foram expandidas e aplicadas em problemas de otimização com muitos objetivos.

Este capítulo apresenta os dois *frameworks* propostos, além das diferentes abordagens de alimentação de *surrogate*. Primeiro é apresentado o *framework* para alimentação de *surrogates* aplicado a problemas mono-objetivos. Junto a esse primeiro *framework* são explicadas as abordagens de alimentação de *surrogate* proposta para problemas mono-objetivos. Em seguida é apresentado o *framework* de alimentação de *surrogate* aplicado a problemas de otimiação com muito objetivos, assim como as abordagens de alimentação de *surrogate* para problemas com muitos objetivos.

3.1 *Framework* para alimentação de *surrogate* aplicado a problemas de otimização mono-objetivos

O *framework* aqui proposto é representado por duas classes de algoritmos trabalhando em conjunto. Esses algoritmos são representados em módulos distintos, o módulo jMetal consiste em algoritmos de otimização e o módulo de aprendizado de máquina composto por algoritmos de aprendizado, conforme ilustrado na Figura 3.

O módulo do jMetal dispõe da classe principal, a qual é responsável pela inicialização

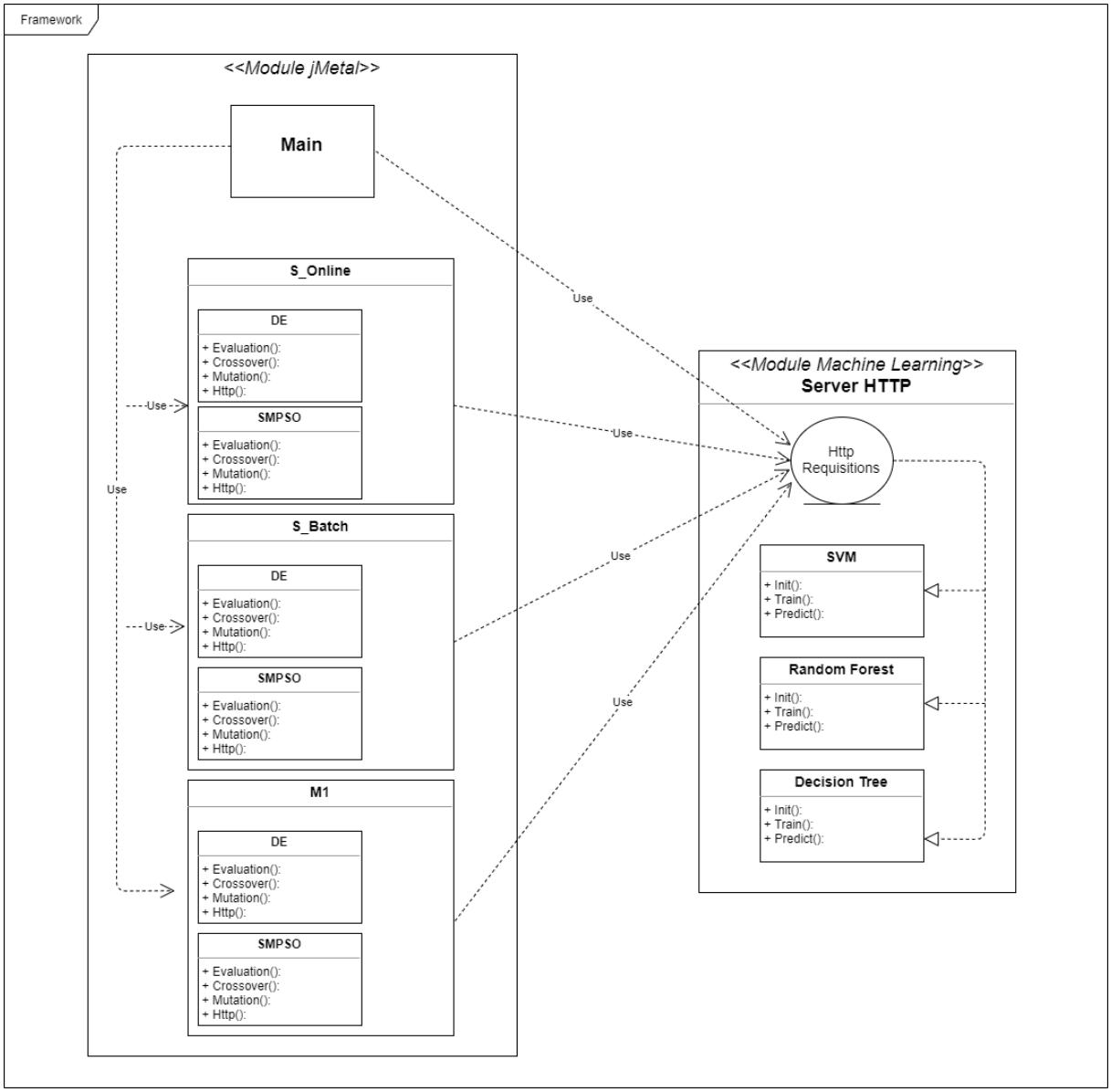

Figura 3 – *Framework* aplicado a problemas mono-objetivos

do algoritmo de aprendizagem de máquina e execução das metodologias de alimentação do *surrogate*. Além da classe principal o módulo jMetal é composto pelas abordagens de alimentação do *surrogate*, S_Online, S_Batch. Além desses dois modelos de alimentação proposto pelos autores o *framework* também implementa o modelo de alimentação M1 presente na literatura. As abordagens de alimentação empregam dois diferentes AE representados pelas classes DE e *Speed-constrained Multi-objective PSO* - (SMPSO). Nas classes dos AE observa-se métodos comuns entre algoritmos evolutivos, como mutação, crossover e avaliação (método usado para avaliar a população). Todas as implementações realizadas no modulo jMetal foram feitas usando a linguagem de programação Java.

Além desses métodos foi implementado um novo método, Http (método criado pois os algoritmos de aprendizagem de máquina foram implementados no *python*), o qual tem como

finalidade realizar a comunicação com o módulo de aprendizagem de máquina. O módulo de aprendizagem de máquina é composto de um módulo que recebe as requisições Http provenientes do módulo jMetal (*Http Requisitions*). Esse módulo pode ser considerado o principal, pois todas as atividades realizadas no módulo aprendizagem de máquina passa primeiro por ele. Além do módulo de requisições existem as classes dos algoritmos de *machine lerning* que são empregados no *framework*, SVM, *Random Forest* e *Decision Tree*. Nessas classes existem métodos comuns entre as classes como, inicialização, treino e predição.

O fluxo do *framework* em geral ocorre da seguinte forma: a classe principal inicializa o algoritmo de aprendizagem de máquina. Além disso ela escolhe e executa a abordagem de alimentação do *surrogate*. Nas abordagens de alimentação a cada nova população gerada o método de avaliação da população é acionado. O método de avaliação da população automaticamente aciona o método *Http*, o qual envia as soluções para o módulo aprendizagem de máquina, ou então guarda as soluções em um arquivo, dependendo do modelo de alimentação. No módulo aprendizagem de máquina o *surrogate* é treinado ou então usado para predizer os valores objetivos das soluções recebidas.

3.1.1 Metodologias de alimentação de *surrogate* aplicado a problemas mono-objetivos

As duas abordagens alimentação do *surrogate* sugeridas nesse subseção são baseadas na rotina de treinamento dos algoritmos de aprendizagem de máquina, *online* e *batch*, as quais foram denominadas de *Surrogate* Online (S_Online) e *Surrogate* Batch (S_Batch). A abordagem S_Online treina o *surogate* à medida que a busca é realizada, a cada atualização da população do algoritmo evolutivo o *surrogate* é treinado. Já na abordagem S_Batch as soluções são armazenadas em um arquivo e dado um valor de avaliações da função objetivo *t_max* o *surrogate* é treinado. A principal diferença entre as duas aboradagens está no processo de treinamento, sendo que na abordagem S_Online o *surrogate* é treinado a cada atualização da população e na abordagem S_Batch o treinamento do *surrogate* ocorre uma única vez.

Na figura 4 é apresentado o fluxograma da abordagem designada de *Surrogate Online*. Essa abordagem está constituida dos seguintes procedimentos, *inicialização da população*: são gerados os conjuntos de soluções iniciais denominadas de população. Após isso é aplicada a etapa de *Processos AE*, que são os procedimentos comuns em algoritmos evolutivos, como mutação, *crossover*, dentre outros. Na etapa seguinte, *Avaliação da população*: a população gerada na etapa anterior é avalida pela função objetivo original. Posterior a essa etapa, o *surrogate* é treinado, *treinamento de surrogate*.

Os passos ligados a etapa *Processos AE* até a etapa *treinamento do surrogate* (Passo 01 à Passo 04) são realizados até que seja atingido o número máximo de avaliações da função objetivo *t_max*. Quando esse valor é atingido, o *surrogate* deixa de ser treinado e passa a ser

usado como substituto da função objetivo, assim na etapa da *Avaliação da população* (Passo 06) não será mais usado a função objetivo real e sim o *surrogate* para predizer o valor do *fitness*, esses procedimentos (Passo 05 até Passo 07) são executados até a condição de parada ser atingida.

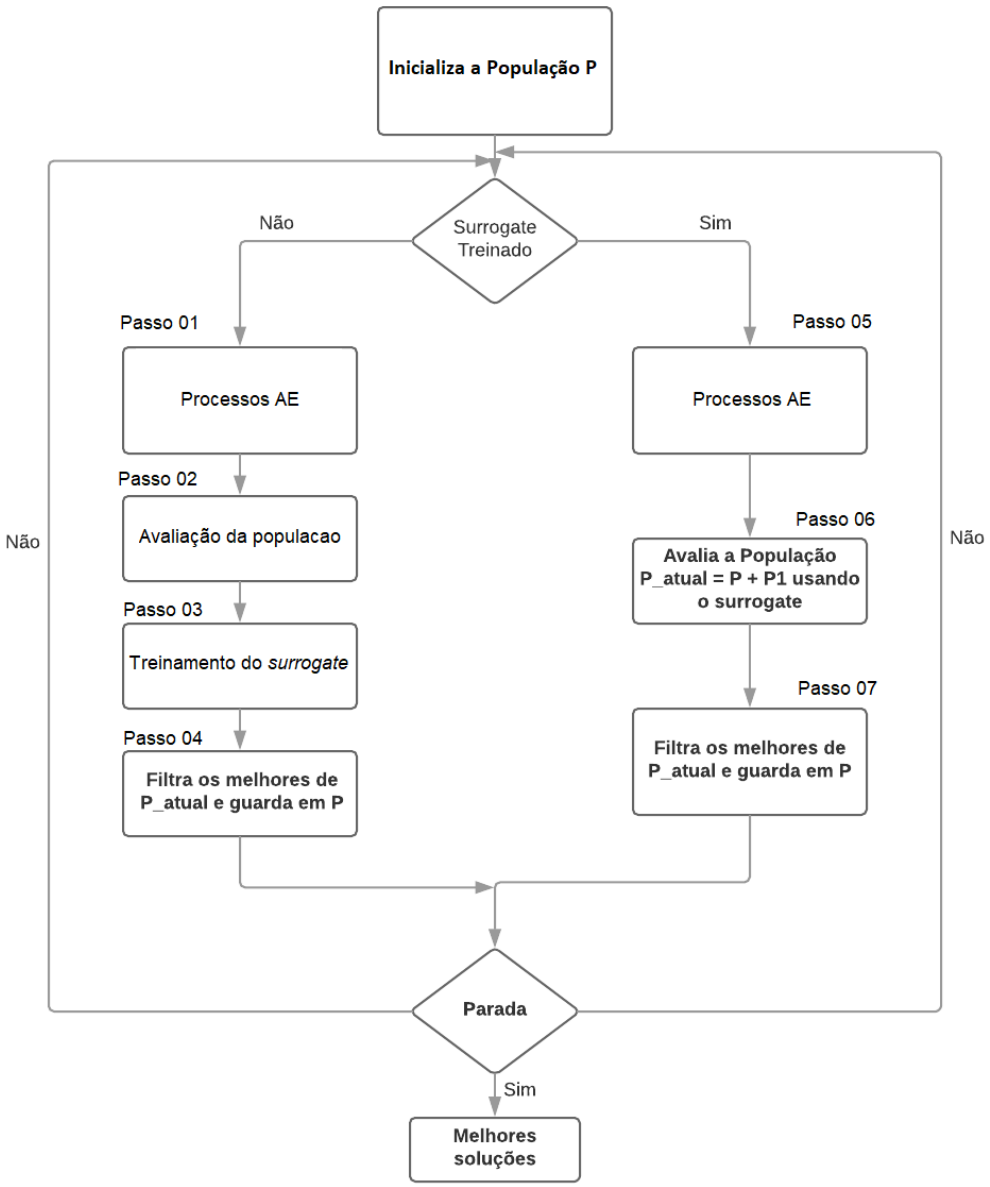

Figura 4 – Fluxograma da metodologia de treino do *surrogate* na forma online

Na outra abordagem proposta é usado o conceito do treinamento do tipo *batch* (todo o conjunto de treinamento deve estar presente no momento em que ocorre o treinamento do modelo) das técnicas de aprendizagem de máquina. O fluxograma da figura 5, ilustra passo a passo os procedimentos realizados nessa abordagem. Essa abordagem tem um fluxo parecido com a abordagem S_online. Nela é realizada a etapa da *inicialização*: todas as entradas são inicializadas, sendo gerado um conjunto de soluções o qual é designado como a população inicial. Após isso é aplicada e etapa de *processos AE*: procedimentos comuns em algoritmos evolutivos,

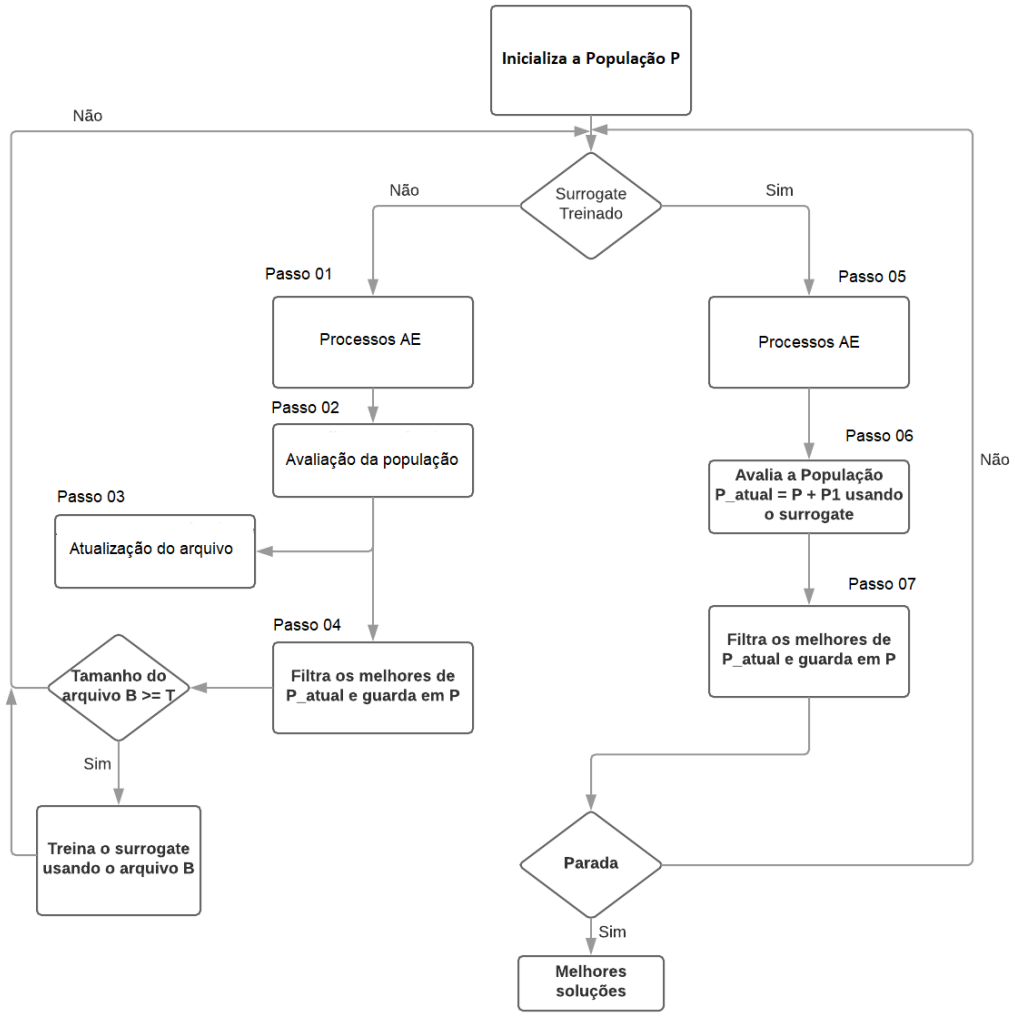

Figura 5 – Fluxograma da metodologia de treino do *surrogate* usando um arquivo, método conhecido como *batch* na área de aprendizagem de máquina

como exemplo tem-se mutação, *crossover*, dentre vários outros. Na etapa seguinte, *Avaliação da população*: a população gerada na etapa anterior é avaliada pela função objetivo original. Posterior a isso ocorre a *Atualização do arquivo*: as soluções da população são adicionadas a um arquivo, após isso é filtrado os melhores elementos da população (Passo 04) e o fluxo retorna para a etapa *processos AE*.

Os passos *processos AE* até *arquivo* (Passos 01 à Passo 04) são executados até que o número máximo de avaliações da função seja antingido. Após isso, o *surrogate* é treinado usando as soluções presentes no arquivo. Após o treinamento do *surrogate*, este passa a substituir a função objetivo real do problema na etapa *avaliação da população* (Passo 06). Os processos executados após o treinamento do *surrogate* (Passos 05 até Passo 07) são realizados até a condição de parada ser atendida.

As duas abordagens S_Online e S_Batch apresentam dois parâmetros de configurações

que irão impactar em seu desmpenho, são eles o *t_max*, que corresponde a quantidade de avaliações da função objetivo usada para treinar o *surrogate* e a condição de parada, *t_parada*, número máximo de avaliações realizado pela função objetivo ou valor de *fitness* abaixo de um valor específico.

A terceira e última abordagem implementa foi a abordagem M1, proposta por (ROY; HUSSEIN; DEB, 2017). Ela usa o processo de *Latin Hypercube Sampling* (LHS) (BEACHKOFSKI; GRANDHI, 2002) para apenas inicializar a população inicial. Nesta são executadas uma sequência de passos com o objetivo de treinar o *surrogate* e otimizar o problemas encontrando as melhores soluções.

Para um melhor entendimento, os passos executados na abordagem M1 encontram-se dispostos no fluxograma apresentado na figura 6. A partir do fluxograma é visível que o treinamento é realizado a cada *t* execução. Ou seja, logo que inicializado a população, é realizada a avaliação dessa população e treinado o *surrogate*. Na etapa seguinte é executado um algoritmo evolutivo para explorar o espaço de soluções, à procura das melhores soluções, usando como função objetivo o próprio *surrogate*. A cada *t* execuções a população usada para realizar o treinamento do *surrogate* inicial é atualizada com a população retornada pelo algoritmo de otmização, e o modelo *surrogate* é retreinado.

Essa abordagem consite em treinar um *surrogate* e por intermédio do algoritmo evolutivo explorar um novo conjunto de soluções, pegando as melhores soluções desse processo e usando para atualizar o *surrogate*, dessa forma são sempre usadas as melhores soluções encontradas a cada *t* execuções no processo de treino do *surrogate*.

A abordagem M1 apresenta o parâmetro *t* como uma especie de controlador, uma vez que ele vai determinar a quantidade de execuções realizadas para realizar novamente o treino do *surrogate*, e a medida que usa-se um *t* pequeno a abordagem irá atualizar o *surrogate* com uma menor diversidade de soluções, já um *t* alto, implica que o *surrogate* será treinado com uma maior aleatoriedade de soluções (ROY; HUSSEIN; DEB, 2017).

3.2 *Framework* de alimentação do *surrogate* aplicado a problemas de otimização com muitos objetivos

Aqui é proposto um novo *framework*, o qual tem como principal carcterística abranger problemas com muitos objetivos. Isso torna necessário o uso de mais de um modelo de aprendizagem, necessariamente, um modelo de aprendizagem para cada função objetivo do problema. Essa é a principal diferença em relação ao *framework* da seção 3.1.

Além disso, o *framework* implementa a função KKTPM empregada por (ROY; HUSSEIN; DEB, 2017). Essa abordagem tem como característica identificar o grau de dominação de uma solução. Essa métrica faz um redimensionamento dos *m* valores das funções objetivo para um

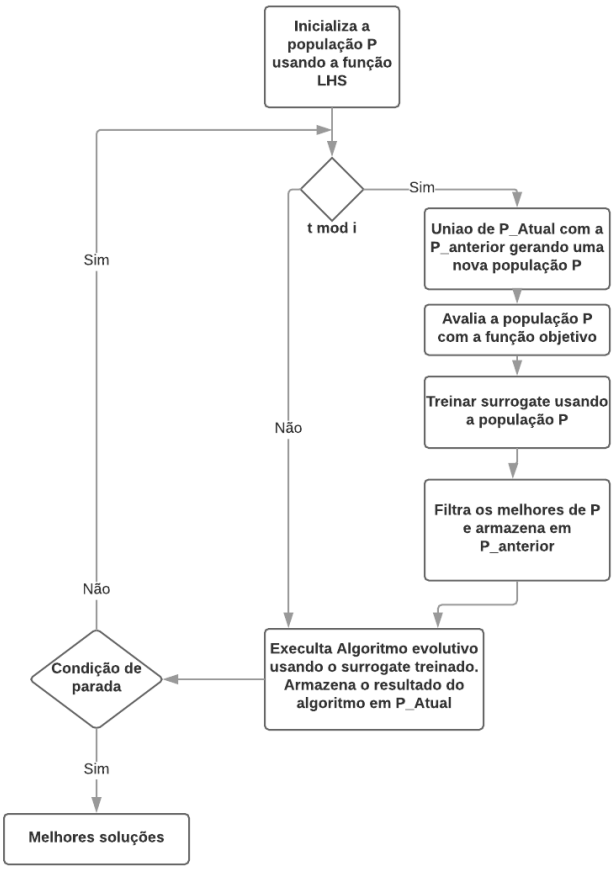

Figura 6 – Fluxograma da metodologia M1 proposta por (ROY; HUSSEIN; DEB, 2017)

único valor, possibilitando assim o uso de um único modelo de aprendizagem.

O *framework* proposto é representado por duas classes de algoritmos trabalhando em conjunto. Esses algoritmos são representados em módulos distintos, o módulo jMetal consiste em algoritmos de otimização e o módulo de aprendizado de máquina composto por algoritmos de aprendizado, conforme ilustrado na Figura 7.

O módulo do jMetal dispõe da classe principal, a qual é responsável pela inicialização do algoritmo de aprendizagem de máquina e execução das metodologias de alimentação do *surrogate*. Além da classe principal o módulo jMetal é composto pelas abordagens de alimentação do *surrogate*, S_MOyO, S_MByO e S_One.

As abordagens de alimentação S_MOyO e S_MByO empregam o algoritmo evolutivo NSGA-II representado pela classe NSGA-II, enquanto a abordagem de alimentação S_One usa o algoritmo SMPSO, representado pela classe SMPSO. Nas classes dos AE observa-se métodos comuns entre algoritmos evolutivos, como mutação, crossover e avaliação (método usado para avaliar a população). Além desses métodos foi implementado um novo método, Http, o qual tem como finalidade realizar a comunicação com o módulo de *machine learning*. Na classe S_One existe um método diferente das demais, o método kktpm(). Ele foi implementado baseado no

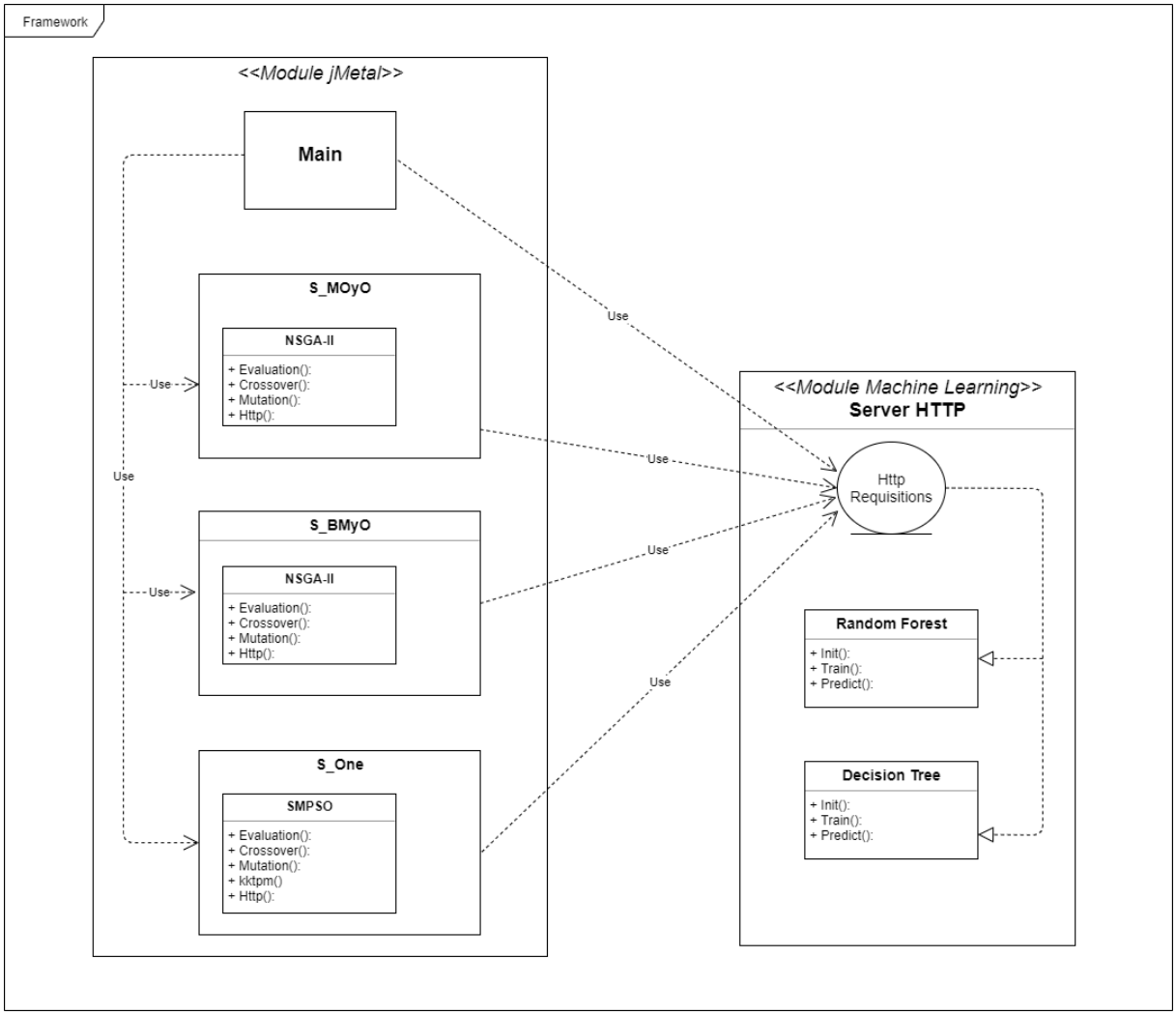

Figura 7 – *Framework* aplicado a problemas com muitos objetivos

trabalho de (DEB et al., 2018; ROY; HUSSEIN; DEB, 2017; DEB et al., 2017; HUSSEIN; DEB, 2016) e tem como finalidade permitir o uso de um único modelo de aprendizagem, independente da quanidade de funções objetivos.

O módulo de *machine learning* é composto de um módulo que recebe as requisições Http provenientes do módulo jMetal (*Http Requisitions*). Esse módulo pode ser considerado o principal, pois todas as atividades realizadas no módulo *machine learning* passa primeiro por ele. Além do módulo de requisições existem as classes dos algoritmos de *machine lerning* que são empregados no *framework*, SVM, *Random Forest* e *Decision Tree*. Nessas classes existem métodos comuns a essas classes de algoritmos como inicialização, treino e predição.

O fluxo do *framework* em geral ocorre da seguinte forma: a classe principal inicializa o algoritmo de aprendizagem de máquina. Além disso ela escolhe e executa a abordagem de alimentação do *surrogate*. Maiores detalhes do funcionamento das abordagens de alimentação são abordados na subseção 3.2.1. Nas abordagens de alimentação a cada nova população gerada o método de avaliação da população é acionado. O método de avaliação da população

automaticamente aciona o método *Http*, o qual envia as soluções para o módulo aprendizagem de máquina, ou então guarda as soluções em um arquivo, dependendo do modelo de alimentação. No módulo aprendizagem de máquina os *m surrogates* são treinados ou então usados para predizer os valores objetivos das soluções recebidas.

3.2.1 Metodologias de alimentação de *surrogate* aplicado a problemas de otimização com muitos objetivos

As abordagens *S_Online* e *S_Batch* abordadas em 3.1.1 tem por características ser empregada em problemas mono-objetivos. Aqui são explanadas extenções para esssas duas abordagens, visando o emprego de ambas em problemas de otimização com muitos objetivos.

O conceito de treinamento de ambas as abordagens, *S_Online* e *S_Batch* continuam, o que muda é o número de classificadores empregados em cada abordagem. Quando trabalha-se com problemas mono-objetivos é necessário apenas um classificador (modelo de aprendizagem) para aprender o comportamento da função objetivo. Para problemas com muitos objetivos, é necessário construir um modelo de aprendizagem para cada função objetivo do problema. Assim, quando se depara com um problema que tem *m* funções objetivos é necessário o uso de *m* modelos.

A expansão da abordagem *S_Online* para muitos objetivos é aqui denominada de *S_OMyO*. Nela o treinamento dos classificadores é realizado a cada interação do algoritmo evolutivo. O desenvolvimento desse procedimento é realizado como segue. O primeiro passo é a *inicialização da população*: são geradas os conjuntos de soluções iniciais. Então é feita a *avaliação da população*: a população gerada anteriormente é avalida pelo conjunto de funções objetivos originais, após essa etapa é realizado o treinamento dos modelos de aprendizagem, para isso a população e os valores objetivos para cada função objetivo são usados para treinar os *m* modelos de aprendizagem. A última etapa é o uso de abordagens comuns em algoritmos de evolutivos para, aqui designada de *ProcessosAE*, nessa etapa ocorre a geração de novas soluções candidatas, usando conceitos de mutação, *crossover*. Após esses procedimentos é retomada a fase de *avaliação da população*. Essa rotina é repetida até atingir a condição de parada.

Como ilustrado no Algoritmo 1 a variável *t_max* corresponde ao número máximo de avaliações da função objetivo para treino do *surrogate*, ou seja, ela é responsável por fazer o controle do momento em que os modelos de aprendizagens deixam de ser treinados e são usados para determinar os valores objetivos da população gerada a cada interação. O algoritmo é executado até que o número máximo de avalições da função objetivo seja atingido, determinado pela variável *t_parada*.

A expansão para problemas de otimização com muitos objetivos também foi aplicada na abordagem *S_Batch*, o processo usado foi similar ao da abordagem *S_Batch* para problemas

Algoritmo 1: Método de Alimentação S_OMyO

Entrada: $Objetivos : (f_1, \ldots, f_m)$, t_parada (Número máximo de avaliações da função objetivo), t_max (Número de avaliações da Função objetivo para treino do Surrogate), P_t (População), ρ (tamanho da população), η (variáveis)

1 $P_t \leftarrow \oslash$
2 $\widehat{P}_k \leftarrow IniciaPopulacao(\rho, \eta)$
3 $eval \leftarrow 0$
Saída: P_t
4 **while** $eval \leq t\_parada$ **do**
5 $\quad \widehat{P}_{k+1} = \widehat{P}_k \bigcup P_t$
6 $\quad$ **if** $eval \leq t\_max$ **then**
7 $\quad\quad \widehat{F}^m_{k+1} = f_m(\widehat{P}_{k+1}), \forall m \in \{1, \ldots, m\}$ // avalia população
8 $\quad\quad S^m_{k+1} = CriaeTreinaSurrogate(\widehat{P}_{k+1}, \widehat{F}^m_{k+1})$ // cria ou treina cada modelo de arendizagem individualmente
9 $\quad$ **else**
10 $\quad\quad \widehat{F}^m_{k+1} = S_m(\widehat{P}_{k+1}), \forall m \in \{1, \ldots, m\}$
11 $\quad$ **end**
12 $\quad P_t \leftarrow$ melhores ρ soluções de $\widehat{P}_{k+1}$
13 $\quad \widehat{P}_k = ProcessosAE(\widehat{P}_{k+1})$
14 $\quad eval \leftarrow eval + \rho$
15 **end**
16 **retorna** P_t

mono-objetivos, variando apenas o arquivo usado para armazenar as soluções de entrada e os seus respectivos valores das funções objetivos, assim como o uso de m modelos de aprendizagem. Sendo um modelo para cada função objetivo apresentada pelo problema. Esta abordagem é denominada de S_BMyO.

Na abordagem S_BMyO, Algoritmo 2, primeiro inicializa-se a população e o conjunto de parametros do algoritmo evolutivo. A próxima etapa é a avaliação da população, nessa etapa atualiza-se o arquivo com as soluções de entrada (população atual) e seus respectivos valores de objetivos. A última etapa constitui de abordagens comuns em AE, designada de *ProcessosAE*, nessa etapa podem ocorrer ações comumns dos algoritmos AE, como mutação, *crossover* para a geração de uma nova população. Apos isso retorna-se para a etapa de avaliação da população, esse ciclo é repetido até que a condição de parada seja atingida.

Aqui á variável t_max também indica o momento em que o arquivo é usado para treinar os modelos de aprendizagem. Quando atingido o valor atribuído a t_max são treinados os m modelos e a partir desse momento o algoritmo evolutivo passa a avaliar as novas populações geradas (a cada interação) usando os modelos de aprendizagem para predizer os valores das funções objetivos.

Algoritmo 2: Método de Alimentação *S_BMyO*

Entrada: $Objetivos : (f_1, \ldots, f_m)$, *t_parada* (Número máximo de avaliações da função objetivo), *t_max* (Número de avaliações da Função objetivo para treino do *Surrogate*), P_t (População), ρ (tamanho da população), η (variáveis), $Archive$ (Arquivo onde são armazenadas as soluções e os valores dos objetivos)

$P_t \leftarrow \oslash$
$\widehat{P}_k \leftarrow IniciaPopulacao(\rho, \eta)$
$eval \leftarrow 0$
Saída: P_t
while $eval \leq t\_parada$ **do**
  $\widehat{P}_{k+1} = \widehat{P}_k \bigcup P_t$
  if $eval \leq t\_max\ Or\ size(Archive) \geq t\_max$ **then**
    $\widehat{F}^m_{k+1} = f_m(\widehat{P}_{k+1}), \forall m \in \{1, \ldots, m\}$ // avalia população
    if $size(Archive) \geq t\_max$ **then**
      $S^m_{k+1} = CriaeTreinaSurrogate(Archive+ < \widehat{P}_{k+1}, \widehat{F}^m_{k+1} >)$ // cria ou treina cada modelo de arendizagem individualmente
      $Archive \leftarrow \oslash$
    else
      $Archive \leftarrow Archive+ < \widehat{P}_{k+1}, \widehat{F}^m_{k+1} >$
    end
  else
    $\widehat{F}^m_{k+1} = S_m(\widehat{P}_{k+1}), \forall m \in \{1, \ldots, m\}$
  end
  $P_t \leftarrow$ melhores ρ soluções de $\widehat{P}_{k+1}$
  $\widehat{P}_k = ProcessosAE(\widehat{P}_{k+1})$
  $eval \leftarrow eval + \rho$
end
retorna P_t

A última metodologia desenvolvida para problemas com muitos objetivos é denominada *S_One*. Essa abordagem usa os conceitos aplicados no *framework* M6 apresentados nos estudos de (DEB et al., 2018; ROY; HUSSEIN; DEB, 2017; DEB et al., 2017; HUSSEIN; DEB, 2016). Esse *framework* M6 apresenta como principal característica usar um único modelo de aprendizado para predizer os valores das funções objetivos, mesmo quando se trata de probemas com mais de uma função objetivo.

O M6 usa a métrica *KKT Proximity Measure* (KKTPM) (DEB; ABOUHAWWASH, 2016), para identificar o grau de dominação de uma solução, como resultados são retornados valores dentro do intervalo 0-1. As soluções que pertencem ao conjunto de soluções ótimas apresentam valores próximos a zeros, e soluções que não pertencem ao conjuntos de soluções ótimas apresentam valores mais elevados, próximos a 1, ou seja, o emprego da métrica KKTPM permite redimencionar os m valores das funções objetivos para um único valor, possibilitando assim usar um único modelo de aprendizado.

O *framework S_One* possui uma grande semelhança com o *framework S_OMyO*. Destacando-se que em *S_One* é aplicado a métrica KKTPM no conjunto de soluções de entrada (população). Com isso ter-se-á um único valor representando cada solução de entrada, possibilitando o uso de um único modelo de aprendizagem.

Algoritmo 3: Método de Alimentação *S_One*

Entrada: $Objetivos : (f_1, \ldots, f_m)$, *t_parada* (Número máximo de avaliações da função objetivo), *t_max* (Número de avaliações da Função objetivo para treino do *Surrogate*), P_t (População), ρ (tamanho da população), η (variáveis)

$P_t \leftarrow \oslash$
$\widehat{P}_k \leftarrow IniciaPopulacao(\rho, \eta)$
$eval \leftarrow 0$
Saída: P_t
while $eval \leq t\_parada$ **do**
  $\widehat{P}_{k+1} = \widehat{P}_k \bigcup P_t$
  $\widehat{D}_{k+1} = kktpm(\widehat{P}_{k+1})$, // retorna um único valor para cada solução
  if $eval \leq t\_max$ **then**
    $\widehat{F}^m_{k+1} = f_m(\widehat{P}_{k+1}), \forall m \in \{1, \ldots, m\}$ // avalia população
    $S_{k+1} = CriaeTreinaSurrogate(\widehat{P}_{k+1}, \widehat{D}_{k+1})$ // cria ou treina o modelo de arendizagem
  else
    $\widehat{F}_{k+1} = S(\widehat{P}_{k+1})$
  end
  $P_t \leftarrow$ melhores ρ soluções de $\widehat{P}_k$
  $\widehat{P}_k = ProcessosAE(\widehat{P}_{k+1})$
  $eval \leftarrow eval + \rho$
end
retorna P_t

O Algoritmo 3 explicita passo a passo os procedimentos realizados no framework *S_One*. O seu desenvolvimento é realizado como segue. O primeiro passo é a *inicialização da população*: são geradas os conjuntos de soluções iniciais. Após isso é aplicada a abordagem KKTPM sobre a população, obtendo-se os valores que representam o grau de dominação de cada solução de entrada, $\widehat{D}$. Após isso, ocorre a *avaliação da população*: a população atual é avalida pelo conjunto de funções objetivos originais. Nessa etapa é realizada o treinamento do único modelo de aprendizagem, usando a populçao atual e grau de dominação de cada solução da população.

A última etapa consiste do uso de abordagens características em algoritmos evolutivos para geração de novas soluções candidatas, designada de *ProcessosAE*. Gerada a nova população retorna-se a etapa de *avaliação da população*. Essa rotina é repetida até atingir a condição de parada.

Como nos demais, o framework *S_One* também utiliza uma variável que indica o momento em que o modelo de aprendizagem deixará de ser treinado e passa a ser usado para

predizer o grau de dominação de cada solução, essa variável é denominada t_max.

Todos os *frameworks* S_OMyO, S_BMyO e S_One apresentam dois parâmetros de configurações essenciais para seu desmpenho, são eles *t_max*, quantidade de avaliações da função objetivo usadas para definir o treino do *Surrogate* e a condição de parada, *t_parada*, quantidade de avaliações da função objetivo.

Salienta-se que os frameworks aqui propostos e trabalhados não estão ligados a nenhuma abordagem de aprendizagem de máquina ou algoritmo evolutivo. Eles estão abertos para o uso de diferentes técnicas de AM assim como diferentes AE.

4

Experimentos e Resultados

Esse capítulo aborda os experimentos e os resultados obtidos para as diferentes abordagens de alimentação do *surrogate* propostas e empregadas nesse trabalho. Para avaliar os objetivos estimados foram planejados e executados duas seções de experimentos. Nessas seções leva-se em consideração as classes de problemas de otimização mono-objetivos e com muitos objetivos, assim como as abordagens de alimentação do *surrogate*.

As diferentes configurações dos experimentos, assim como seus resultados, serão apresentados levando em consideração as classes dos problemas de otimização, sendo assim, primeiro é apresentado o conjunto de experimentos para os problemas de otimização mono-objetivos e por conseguinte os experimentos e resultados para os problemas de otimização com muitos objetivos.

4.1 Experimentos e resultados para problemas mono-objetivos

Aqui são apresentadas todas as configurações e metodologias empregadas nos experimentos para problemas mono-objetivos, assim como a discussão dos resultados obtidos.

4.1.1 Planejamento dos experimentos

Seleção de Contexto: o experimento foi *in vitro* e usou problemas de otimização monoobjetivo simulados, de acordo com sua modelagem matemática, usando a linguagem de programação Python e Java, sendo usados cinco problemas diferentes (*Ackley*, *Ellipsoid*, *Griewank*, *Rastrigin* e *Rosembrock*) (SURJANOVIC; BINGHAM,). Além disso, foram utilizados três algoritmos de aprendizagem de máquina na forma de *surrogate* (SVM, Árvore de Decisão e *Random Forest*). Essas técnicas de aprendizagem de máquina foram empregadas na forma de *surrogate* em dois algoritmos evolutivos, *Differential Evolution* (DE) e *Speed-constrained Multi-objective PSO* (SMPSO).

Variáveis Dependentes: Hiperparâmetros dos algoritmos de aprendizagem de máquina.

Variáveis Independentes: Problemas de otimização usados (*Ackley*, *Ellipsoid*, *Griewank*, *Rastrigin* e *Rosembrock*), variáveis de decisão dos problemas (apresentados na Tabela 5) e os próprios algoritmos (SVM, Árvore de Decisão e *Random Forest*).

Projeto do Experimento: Nas abordagens *S_Online* e *S_batch*, supõe-se que o *surrogate* esteja treinado quando o algoritmo obtiver cinquenta mil (50.000) avaliações da função objetivo. Nesse momento, o algoritmo para de usar a função objetivo do problema real e começa a usar o *surrogate* para avaliar as soluções de entrada. Os algoritmos são executados até que cem mil (100.000) soluções de entrada sejam avaliadas. Os primeiros 50.000 são utilizados no processo de aprendizado, além das 50.000 novas soluções geradas pelo algoritmo, agora usando o *surrogate*. A seleção desses números será discutida mais detalhadamente na subseção 4.1.2

Instrumentação: os algoritmos utilizados foram obtidos a partir de bibliotecas provenientes da linguagem de programação Python na versão 3.5.1 (PEDREGOSA et al., 2011) e jMetal (DURILLO; NEBRO, 2011). A execução dos algoritmos foi feita por meio de um computador Dell, com 8Gb de memória RAM, processador Intel i5-3470S (2.9GHz) executando em um sistema operacional Ubuntu 12.04.5 LTS.

4.1.2 Preparação

Dois algoritmos meta-heurísticos, DE e SMPSO, foram aplicados para resolver problemas *benchmark*. Junto a eles foram empregados os modelos de aprendizagem de máquinas SVR, árvore de decisão e *Random Forest* como *surrogate*. Foram criados IDs para identificar as diferentes composições de algoritmos e *surrogate* empregados. Os IDs obedecem à seguinte ordem: nome do algoritmo + abordagem de treinamento + algoritmo de aprendizagem de máquina. Os diferentes IDs para as configurações implementadas estão organizados na Tabela 3. Como critério de parada, cada configuração de algoritmo, apresentada na Tabela 3, foi executada até que cem mil avaliações de funções objetivas fossem alcançadas. Além disso, as versões originais dos algoritmos SMPSO e DE foram executadas sem interferência das abordagens *surrogate*. O algoritmo evolutivo foi executado usando a função objetivo real de cada problema.

Para cada algoritmo foi empregado os hiperparâmetros dispostos na tabela 4. Para os algoritmos evolutivos, SMPSO e DE foi usado os parâmetros padrões dispostos no framework jMetal.

Para os algoritmos de aprendizagem de máquina foram executados alguns exemplos variando os principais parâmetros. Para o SVR variou-se os valores de C (1e0, 1e1, 1e2, 1e3, 1e4), para o *Random Forest* variou-se o *number of estimators* (100, 200, 300, 400, 500) e para o árvore de decisão variou-se o *maximum depth* (200, 300, 400, 500, 1000). A partir dos exemplos executados obteve-se os conjuntos de hiperparâmetros dispostos na tabela 4 como os melhores resultados.

Tabela 3 – Configurações *Surrogate*

Configuração	Codinome
DE + S_Online + SVR	DEOnlineSVR
DE + S_Batch + SVR	DEBatchSVR
DE + M1 + SVR	DEM1SVR
DE + S_Online + RF	DEOnlineRF
DE + S_Batch + RF	DEBatchRF
DE + M1 + RF	DEM1RF
DE + S_Online + AD	DEOnlineAD
DE + S_Batch + AD	DEBatchAD
DE + M1 + AD	DEM1AD
SMPSO + S_Online + SVR	SMPSOOnlineSVR
SMPSO + S_Batch + SVR	SMPSOBatchSVR
SMPSO + M1 + SVR	SMPSOM1SVR
SMPSO + S_Online + RF	SMPSOOnlineRF
SMPSO + S_Batch + RF	SMPSOBatchRF
SMPSO + M1 + RF	SMPSOM1RF
SMPSO + S_Online + AD	SMPSOOnlineAD
SMPSO + S_Batch + AD	SMPSOBatchAD
SMPSO + M1 + AD	SMPSOM1AD

Tabela 4 – Parâmetros dos algoritmos

Algoritmo	**Parêmetros**
SMPSO	*population size* = 100, *mutation probability* = 0.05 , *mutation distribution* = 20
DE	*population size* = 100, *crossover*: cr = 0.5, f = 0,5 *and variant* = rand/1/bin
Random Forest	*number of estimators* = 200, *minimum samples split* = 2 , *random state* = 0, *criterion of error* = mse
SVR	*kernel* = rbf, C = 1e3, *gamma* = 0.1, *tolerance error* = 0.01
Decision Tree	*maximum depth* = 500, *minimum samples split* = 2, *random state* = 0, *criterion of error* = mse

As diferentes configurações de algoritmos listadas na Tabela 3 foram aplicadas em cinco problemas mono-objetivos, *Ackley*, *Ellipsoid*, *Griewank*, *Rastrigin*, *Rosembrock*. Para cada problema foram usados os parâmetros ilustrados na Tabela 5. Todos esses problemas já se encontram implementados no *framework* jMetal, ou então foram implementados no mesmo a partir de sua formulação matemática (SURJANOVIC; BINGHAM,).

4.1.3 Execução

Como apresentado anteriormente, assumiu-se que o *surrogate* é considerado treinado quando o algoritmo atinge cinquenta mil (50.000) avaliações da função objetivo. Para a escolha desse número, inicialmente, diferentes valores foram testados. Essa análise foi realizada como

Tabela 5 – Parâmetros dos Distúrbios

Problema	**Parâmetros**	**Equações**
Ackley	$a = 20, b = 0.2, c = 2\pi, x \in [-32.768, 32.768], f(x) = 0$	$f(x) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} {x_i}^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$
Ellipsoid	$x \in [-65.536, 65.536], f(x) = 0$	$f(x) = \sum_{i=1}^{d}\sum_{j=1}^{i} {x_j}^2$
Griewank	$x \in [-600, 600], f(x) = 0$	$f(x) = \sum_{i=1}^{d-1}\frac{{x_i}^2}{4000} - \prod_{i=1}^{d-1}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Rastrigin	$x \in [-5, 10], f(x) = 0$	$f(x) = 10d + \sum_{i=1}^{d-1}[({x_i}^2 - 10\cos(2\pi x_i)]$
Rosembrock	$x \in [-5.12, 5.12], f(x) = 0$	$f(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2]$

forma de identificar o melhor momento para finalizar o processo de treinamento e iniciar o uso do *surrogate* para avaliar as soluções. O problema escolhido para a anlise foi o Rosembrock, seguindo as configuração ilustradas na Tabela 6, para os algoritmos DE e SMPSO. As configurações escolhidas foram SMPSOOnlineRF e DEOnlineRF.

Tabela 6 – Diferentes valores usados para treino e teste do modelo de aprendizagem

Treino (mil)	**Teste (mil)**	**Valor Máximo (mil)**
10	90	100
20	80	100
30	70	100
40	60	100
50	50	100

Na Figura 8 nota-se que, antes mesmo das duas mil avaliações da função objetivo, o algoritmo evolutivo SMPSO já se encontra em uma região próxima ao valor ótimo como resposta para o problema. Assim, o valor limiar a ser usado teria que ser inferior a duas mil avaliações da função objetivo.

Entretanto ao observar o mesmo limiar quando aplicado ao algoritmo DE, ver Figura 9, nota-se que o algoritmo tende a chegar próximo a valores ótimos quando aproxima-se das cinquenta mil avaliações da função objetivo. Dessa forma, foi adotado como limiar o valor máximo entre os dois mapeados, adotando-se assim, um valor de cinquenta mil avaliações como valor limiar para parar de treinar o algoritmo evolutivo e começar a usá-lo para substituir a função objetivo do problema.

4.1.4 Medida de qualidade

Conforme apresentado inicialmente, as perguntas de pesquisa para essa classe de problemas serão respondidas usando como métrica de qualidade o valor da função objetivo, alcançada pelo algoritmo. Cada configuração na Tabela 3 foi executada vinte vezes, na qual a

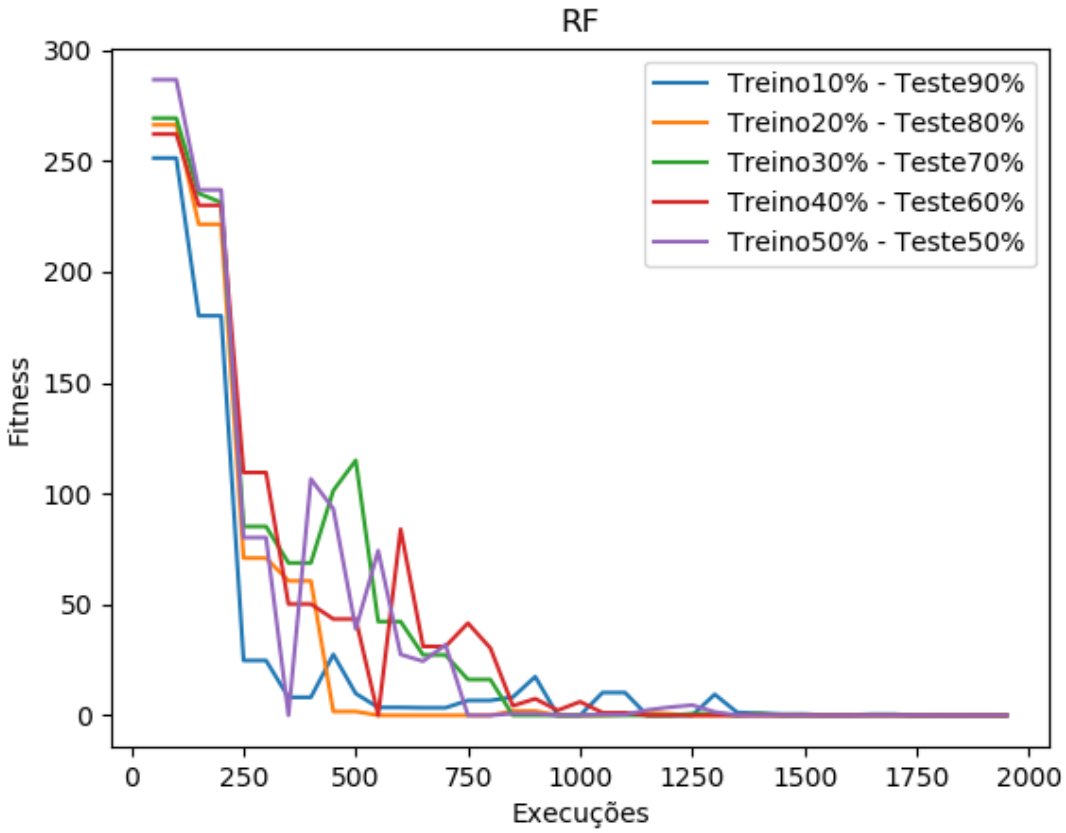

Figura 8 – Diferentes valores de treino do *Surrogate* para a configuração SMPSOOnlineRF

melhor solução para *fitness* foi armazenada. As médias e desvios padrão desses vinte melhores valores de aptidão foram analisados. Além dessa análise, também foi feita uma avaliação da aprendizagem do modelo durante treinamento, assim como o desempenho do modelo quando utilizado para predizer os valores da função objetivo.

Para confirmar ou refutar as hipóteses (hipóteses 1 e 2), foi aplicado o teste estatístico de *Wilcoxon* (DERRAC et al., 2011). Com isso, foi possível identificar as configurações cujos resultados não apresentavam similaridades e, portanto, indicar aquelas que obtiveram os melhores resultados. O critério utilizado para escolher os melhores resultados foi o valor p_value. Para configurações com valor de p_value menor que 0,05, é possível inferir estatisticamente a diferença entre elas.

A partir das inferências feitas nos resultados com base nos testes de hipóteses 1 e 2:

Teste de Hipótese 1

H0: As técnicas de aprendizagem de máquina possuem o mesma eficácia para aprender as funções objetivos de algoritmos evolutivos, quando aplicados a problemas mono-objetivos;

H1: As técnicas de aprendizagem de máquina possuem não possuem o mesma eficácia para aprender as funções objetivos de algoritmos evolutivos, quando aplicados a problemas

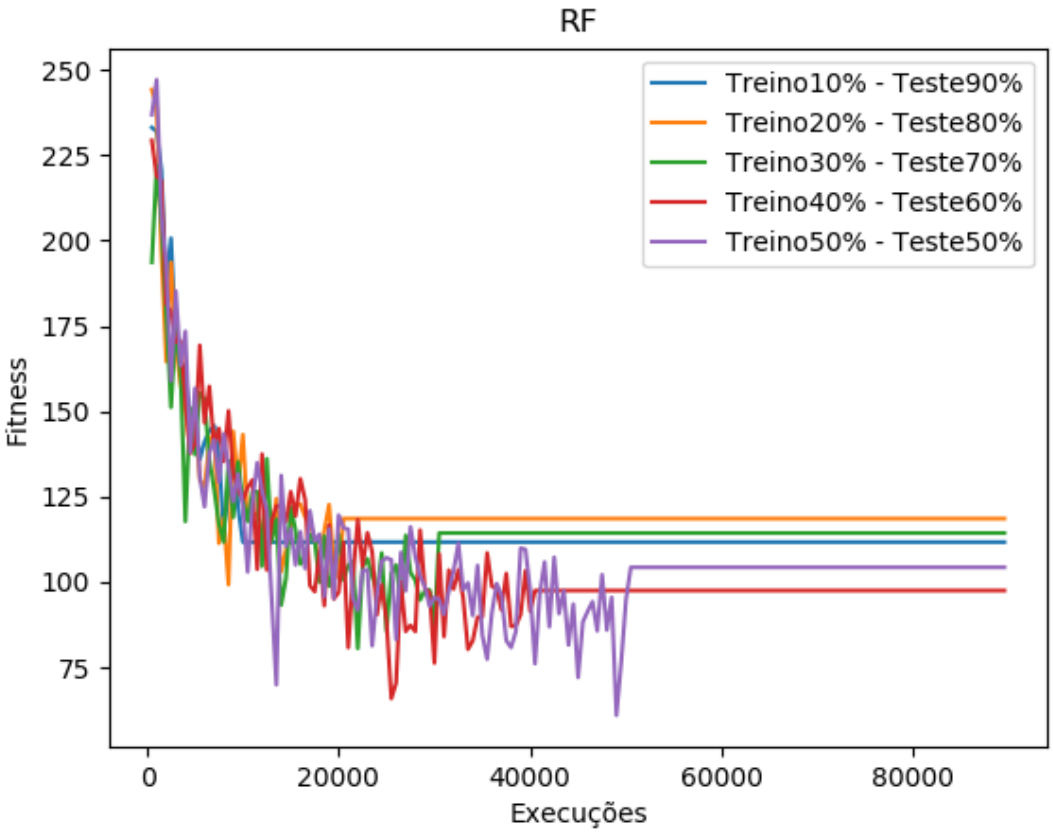

Figura 9 – Diferentes valores de treino do *Surrogate* para a configuração DEnlineRF

mono-objetivos;

Teste de Hipótese 2

H0: Os diferentes modelos de alimentação do *surrogate* proporcionam resultados iguais quando aplicados em algoritmos assitidos por surrogate;

H1: Os diferentes modelos de alimentação do *surrogate* proporcionam resultados diferentes quando aplicado em algoritmos assitidos por *surrogate*;

Estima-se ser possível responder as seguintes perguntas de pesquisas, *Qual a melhor técnica de aprendizagem de máquina para aprender e substituir as funções objetivos de um algoritmo evolutivo aplicado a problemas de otimização?* e *O modelo de alimentação do surrogate influencia no desempenho do algoritmo evolutivo assistido por surrogate?*

A próxima seção apresenta os resultados e a discussão. As tabelas 7 e 8 apresentam os valores das métricas de qualidade. Os resultados destacados em negrito são aqueles em que o teste de *Wilcoxon* não computou diferenças estatísticas, ou seja, não pode-se afirmar que os valores são estatisticamente diferentes. Além disso, a seção apresentará alguns gráficos sobre os resultados obtidos.

4.1.5 Resultados e Discussão

Nessa Seção serão apresentados os resultados para os experimentos iniciais. Os resultados encontram-se dispostos em duas etapas, na primeira etapa são apresentados as médias e seus respectivos desvios padrões para as vinte execuções de cada configuração da Tabela 3. Para esses resultados é feita uma análise identificando as abordagens de *surrogate* que obtiveram os melhores resultados.

Em uma segunda etapa é realizada uma análise que visa identificar o quão próximo está o valor do *fitness* predito pelo *surrogate* em comparação ao valor da função objetivo original.

4.1.5.1 Valores Médios das Execuções

As Tabelas 7 e 8 apresentam a média e desvio padrão das vinte execuções para cada configuração, sendo que, a Tabela 7 ilustra os resultados para as configurações que usam o algoritmo DE e a Tabela 8 os resultados para as confirgurações usando o algoritmo SMPSO.

Em ambas tabelas encontram-se destacados em negrito os melhores resultados, escolhidos tomando como base o retorno do teste estatístico de *wilcoxon*. No teste todas as configurações para cada problema foram confrontadas, exceto a configuração base do algoritmo.

Nessa análise não foi considerado os valores médios da configuração base, que trata-se dos valores obtidos usando os dois algoritmos evolutivos (SMPSO e DE) sem o uso de *surrogate*. Porém esses valores são apresentados com o intuito de realizar um comparativo entre os valores médios da configuração do algoritmo base e das configurações usando *surrogate*, relativisando o quão próximo ou distante esses valores estão.

Aqui é apresentada uma análise mais detalhada em termos dos resultados. Essa análise terá como foco avaliar as diferentes técnicas de aprendizagem de máquina empregadas e será feita sobre os resultados de cada problema, sem priorizar o algoritmo ou abordagem usada.

Como complemento aos resultados visualizados nas Tabelas 7 e 8 serão usados gráficos *bloxplot* dos resultados das vinte execuções de cada configuração. Alguns resultados, para determinadas configurações, apresentam uma grande oscilação nos valores médios, difucultando o entendimento dos gráficos. Essas configurações com valores elevados foram removidas para melhor compreensão.

Quando explorado os resultados obtidos para o problema *Ackley*, observa-se que as técnicas de AD e RF obtiveram os melhores desempenhos, possuindo os melhores resultados nas configurações do tipo *online* e *batch* associada ao algoritmo SMPSO e também obtivem os melhores resultados nas abordagens do tipo *online* associada ao algoritmo DE. Ao observar o valor base em detrimento dos valores das melhores configurações é possível notar uma proximidade entre estes, isso implica que as técnicas destacadas como as melhores obtiveram sucesso em aprender o comportamento da função objetivo do problema.

Tabela 7 – Resutados médio das diferentes configurações de *surrogate* para o algoritmo DE

	Ackley	Ellipsoid	Rastrigin	Rosembrock	Griewank
Base	6.2e-10 ± 1.5e-10	4.2e-19, 2.0e-19	51.616 ± 6.618	13.288 ± 0.531	2.1e-12, 8.6e-12
DEOnlineSVR	**0.0003 ± 4.573e-05**	0.704 ± 0.885	121.593 ± 11.081	165.846 ± 22.452	0.045 ± 0.083
DEBatchSVR	0.0383 ± 0.0174	1933.231 ± 579.901	**65.263 ± 6.57**	10725.200 ± 3854.97	45.920 ± 15.363
DEM1SVR	2.894 ± 0.1167	27.106 ± 2.605	198.200 ± 39.0577	196.851 ± 18.524	0.245 ± 0.0169
DEOnlineRF	**0.00032, 6.4e-05**	**4.5e-05, 5.5e-05**	124.093 ± 5.475	**15.608 ± 0.147**	**0.035 ± 0.084**
DEBatchRF	0.0006, 5.98e-05	**0.0001 ± 4.6e-06**	85.794 ± 5.667	**15.550 ± 0.169**	**0.0098 ± 0.0179**
DEM1RF	3.212 ± 0.099	**6.5e-07 ± 2.2e-07**	134.936 ± 7.332	216.110 ± 15.553	0.193 ± 0.0164
DEOnlineAD	**0.00035 ± 5.9e-05**	**7.2e-07 ± 1.8e-07**	**66.456 ± 6.498**	**15.347 ± 0.177**	**0.009 ± 0.021**
DEBatchAD	0.0006 ± 6.3e-05	**7.2e-05 ± 5.8e-05**	**65.555 ± 8.804**	**15.381 ± 0.187**	**0.0027 ± 0.004**
DEM1AD	3.076 ± 0.111	18.293 ± 2.957	117.814 ± 8.488	169.892 ± 23.461	0.173 ± 0.0268

A partir da Figura 10 nota-se visualmente o comportamento de todas as configurações. É possível identificar que os resultados obtidos para as técnicas AD e RF associadas a abordagem *batch* obteve resultados próximos aos melhores resultados, ver Tabelas 7 e 8 e Figuras 10-B e 10-D, porém, de acordo com o teste estatístico, existe diferença estatística entre os resultados.

Partindo para o problema *Ellipsoid*, foi identificado que as técnicas AD e RF obtiveram os melhores resultados quando associadas às abordagens *online* e *batch* para ambos os algoritmos, sendo que a técnica RF também obteve um bom resultado para a abordagem M1 em conjunto com o algoritmo DE. A Figura 11 ilustra visualmente esses resultados inferidos a partir do teste estatístico. A partir dela é nítido que as configurações envolvendo as técnicas AD e RF possuem em geral o melhor conjunto de médias.

Embora as configurações que usam as abordagens M1 associadas às técnicas AD e RF possuam resultados próximos a zero, essas configurações possuem resultados estatisticamente diferentes quando comparados com os melhores resultados.

Quando comparado os valores obtidos para as configurações computadas como as melhores com o valor base, apenas usando o algoritmo evolutivo, nota-se que os valores estão próximos, sentenciado o êxito das técnicas em substituir a função objetivo.

As Figuras 12 e 13 ilustram os resultados obtidos para o problema *Rastrigin*. Nela é explícito que as configurações usando as técnicas AD e RF associadas as abordagens *online*

Tabela 8 – Resutados médio das diferentes configurações de *surrogate* para o algoritmo SMPSO

	Ackley	Ellipsoid	Rastrigin	Rosembrock	Griewank
Base	4.4409e-16 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	13.4321 ± 4.5144	0.0 ± 0.0
SMPSOOnlineSVR	2.4576 ± 0.6780	244.0288 ± 102.1398	283.9736± 48.7185	823.0879 ± 217.5078	56.4277 ± 12.3163
SMPSOBatchSVR	10.8890 ± 2.4750	3608.1035 ± 1364.6196	616.1877 ± 136.3645	7795.3570 ± 2772.3648	20.6150 ± 4.7586
SMPSOM1SVR	0.0171 ± 0.9911	377.6982 ± 140.0080	386.3096± 32.7341	1489.2421 ± 448.8067	46.2318 ± 11.9469
SMPSOOnlineRF	**9.5e-08 ± 2.9e-07**	**6.7e-09 ± 1.17e-08**	**1.4e-08 ± 2.5e-08**	**14.0709 ± 6.5889**	**6.8e-09 ± 2.5e-08**
SMPSOBatchRF	**0.0001 ± 6.3e-06**	**8.1e-05 ±** 5.4e-06	**7.4e-05 ± 6.5e-06**	**17.3806 ± 0.5427**	**7.5e-05 ± 6.5e-06**
SMPSOM1RF	0.4329 ± 0.7096	0.0751 ± 0.1596	0.9827 ± 2.2711	1572.8443 ± 466.0352	**0.0040 ± 0.0068**
SMPSOOnlineAD	**4.4e-16 ± 0.0**	**0.0 ± 0.0**	**0.0± 0.0**	**14.465 ± 6.1100**	**0.0 ± 0.0**
SMPSOBatchAD	**9.3e-05 ± 2.0e-05**	**5.5e-05 ± 3.7e-05**	**7.4e-05 ± 1.4e-05**	**13.7666 ± 6.8932**	**6.8e-05 ± 1.6e-05**
SMPSOM1AD	0.7087 ± 0.8075	0.3714 ± 0.4583	1.2516 ± 2.8896	17.8661 ± 4.0990	0.03750 ± 0.0669

e *batch* obtiveram os melhores resultados para o algoritmo SMPSO. Enquanto a técnica AD associada às abordagens *online* e *batch* para o algoritmo DE obteve os melhores resultados, juntamente com a técnica SVM associada à abordagem *batch*.

A confirmação desses resultados é feita a partir dos testes estatísticos aplicados para os resultados desse problema, ver Tabelas 7 e 8. Também é possível constatar que os valores médios obtidos para a técnica RF quando associada à abordagem *online* obteve resultado não tão distante dos melhores resultados. Todos os resultados computados como melhores para esse problema encontram-se acima do resultado médio do experimento base, porém existe uma proximidade entre estes, ratificando que as técnicas vencedoras são capazes de aprender a função objetivo.

Para o problema *Rosembrock* as técnicas AD e RF associadas às abordagens *online* e *batch* foram as que apresentaram os melhores resultados, para ambos os algoritmos. Esse resultado pode ser visualiado de forma clara nos gráficos ilustrados na Figura 14, assim como nas Tabelas 7 e 8.

Ao analisar esses resultados tomando como referência os resultados para a configuração base é possível identificar uma proximidade entre eles, isso salienta a capacidade das técnicas AD e RF em aprender a função objetivo. Para o último problema analisado, *Griewank*, nota-se que as técnicas AD e RF continuaram a obter os melhores resultados. Fator validado pelos gráficos da Figura 15.

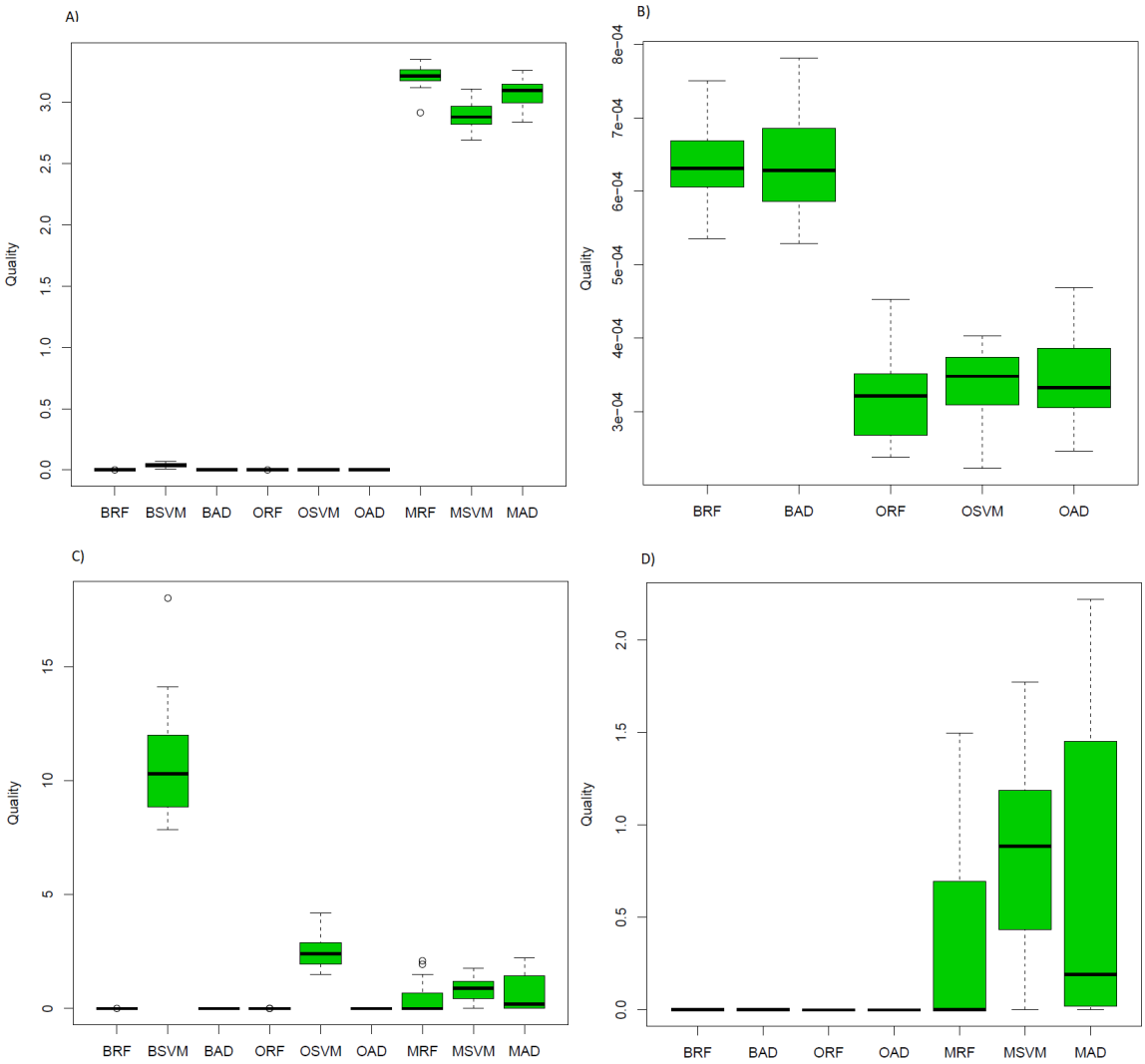

Figura 10 – Gráficos *bloxplot* para as diferentes configurações aplicadas as problema *Ackley*. A) Todas as configurações *surrogate* associada ao algoritmo DE, B) melhores resultados para as configurações associada ao algoritmo DE, C) Configurações de *surrogate* associadas ao algoritmo SMPSO e D) melhores configurações de *surrogate* associadas ao algoritmo SMPSO

Nesse problema também foi ratificado que os valores obtidos para as melhores configurações possuem uma proximidade com o valor base, com isso, ratifica-se a capacidade dessas técnicas em aprender o comportamento de diferentes funções objetivos. Diante dos resultados aqui expostos, nota-se que dentre as técnicas avaliadas, os algoritmos AD e RF obtiveram os melhores conjuntos de resultados para todos os cinco problemas submetidos.

Estima-se que esse resultado esteja diretamente associado à capacidade de generalização de ambas as técnicas, AD e RF. Essa capacidade vem do poder de particionar o espaço de aprendizagem. Em ambas as técnicas o espaço de aprendizagem é dividido em N partições menores que são então classificadas.

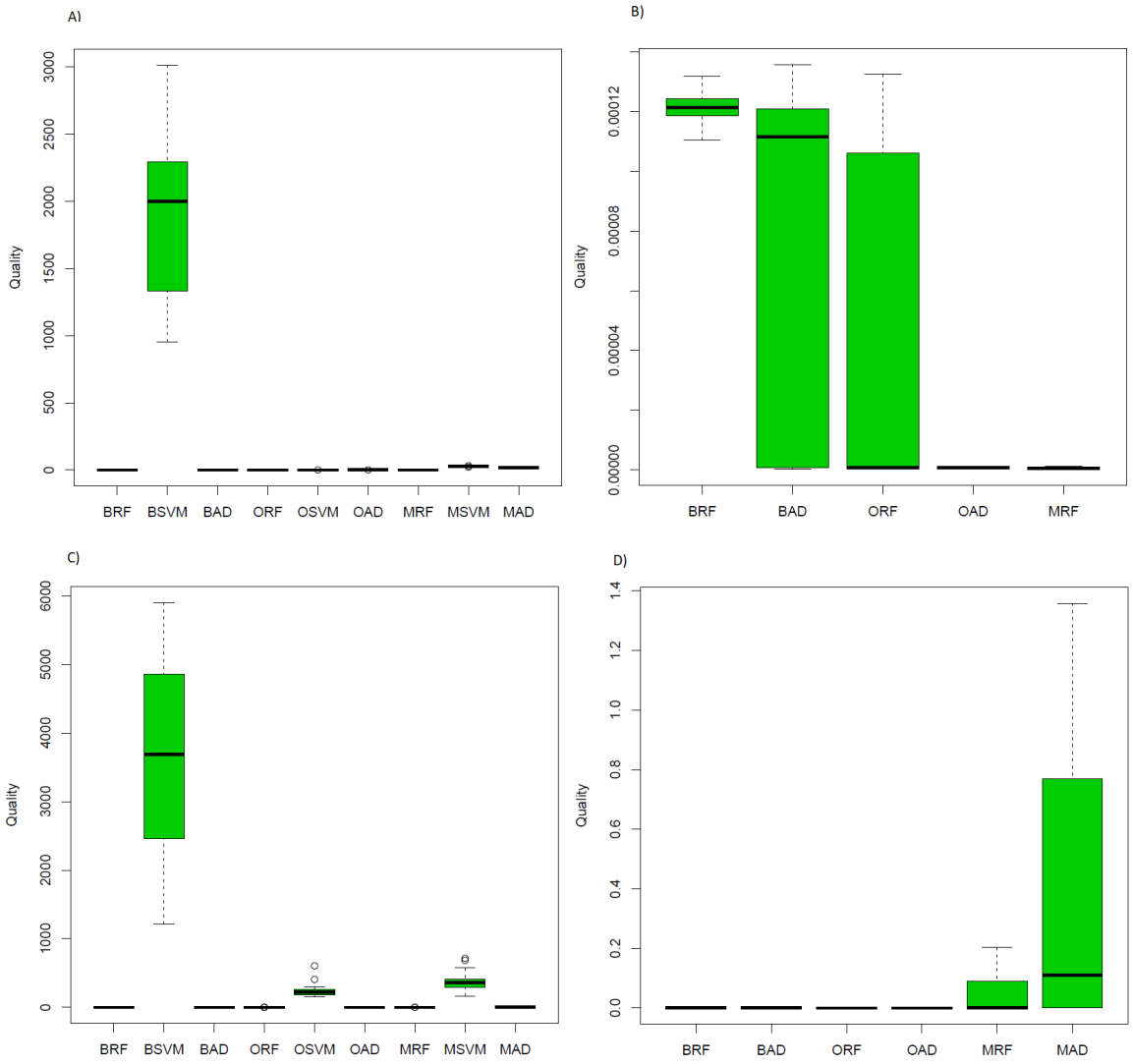

Figura 11 – Gráficos *bloxplot* para as diferentes configurações aplicadas ao problema *Ellipsoid*. A) Todas as configurações *surrogate* associada ao algoritmo DE, B) melhores resultados para as configurações associada ao algoritmo DE, C) Configurações de *surrogate* associadas ao algoritmo SMPSO e D) melhores configurações de *surrogate* associadas ao algoritmo SMPSO

Uma vez que os algoritmos evolutivos avaliam milhares de soluções para encontrar as melhores soluções, o processo de aprendizagem das técnicas AD e RF irá usar essas milhares de soluções, particionando-as em milhares de pequanas partições, as quais são rotuladas. Esses modelos mostraram-se pomissores, com um bom comportamento na perspectiva de aprender o comportamento da função objetivo.

Em contrapartida, nesse estágio, não é possível identificar se a abordagem empregada para o treinamento da técnica de aprendizagem de máquina influenciou nos resultados. Para isso será feita uma análise a seguir que tem como finalidade averiguar o comportamento do aprendizado da técnica de aprendizagem de máquina independentemente da abordagem usada.

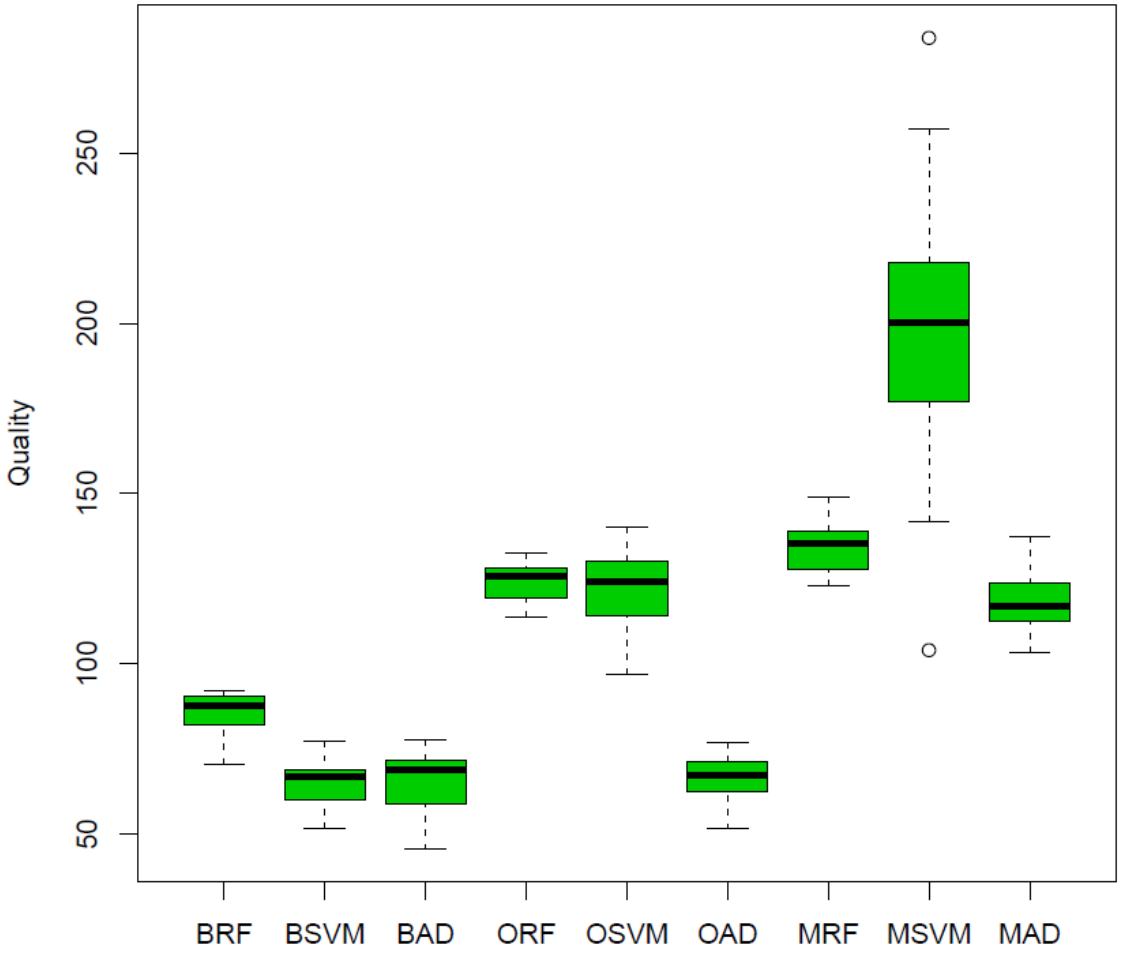

Figura 12 – Gráficos *bloxplot* para as diferentes configurações associadas ao algoritmo DE aplicadas ao problema *Rastrigin*.

4.1.5.2 Curva de aprendizagem

Aqui serão apresentadas quais são as melhores configurações, assim como, o comportamento de cada configuração em relação aos resultados do algoritmo base com o decorrer da busca. Mediante a isso foram construidos gráficos que especificam esse comportamento. Para cada avaliação de *fitness* (cem mil para cada configuração) foram armazenados seus valores, e, diante disso, foi gerado um gráfico que apresenta o comportamento do *fitness* das primeiras avaliações às últimas.

É esperado, que nas avaliações iniciais o valor do *fitness* esteja alto, e, ao decorrer da busca, esse valor decaia (uma vez que são problemas de minimização), chegando próximo a zero. Para uma melhor apresentação dessa avaliação foi construído um gráfico para cada problema, agrupado de acordo com a técnica de aprendizagem de máquina.

As Figuras 16 à 23 apresentam o comportamento da busca a medida que ela é realizada. Esse gráfico representa o número de avaliações da função objetivo vs valores de *fitness*. A cada

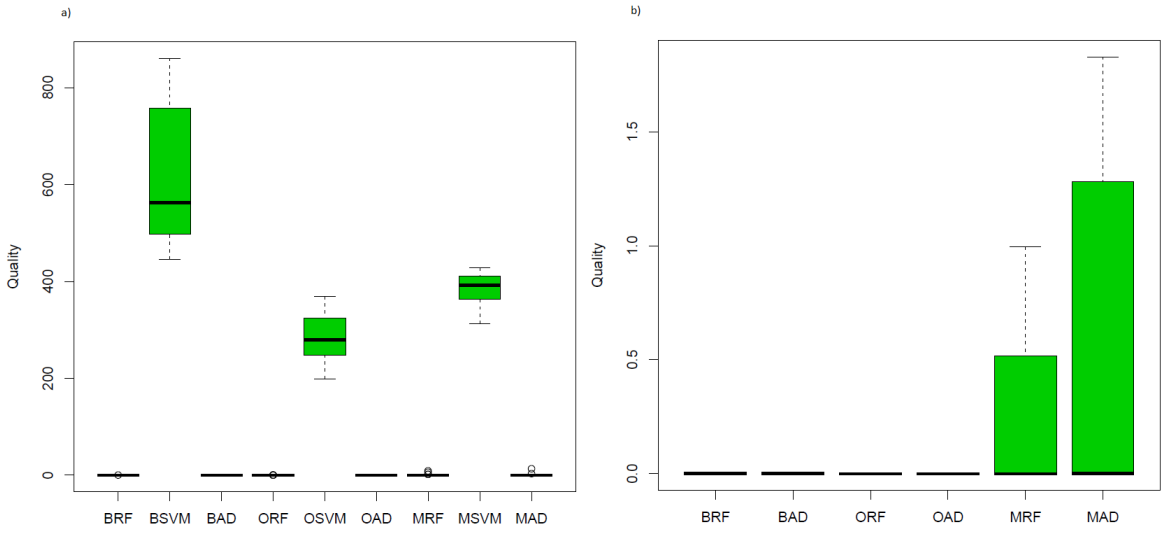

Figura 13 – Gráficos *bloxplot* para as diferentes configurações associadas ao algoritmo SMPSO aplicadas ao problema *Rastrigin*. A) todas as configurações e B) melhores configurações

mil avaliações da função objetivo foi coletado o menor valor de *fitness*, gerando assim um ponto do gráfico. Inicialmente, quando o número de avaliações da função objetivo é pequeno, nota-se que o valor de *fitness* da função objetivo é elevado. A medida que a busca evolui, e um maior numero de avaliações é realizado, o valor do *fitness* decai, tendendo a chegar ao valor ótimo, isso ocorre quando já foram avaliados um exaustivo número de soluções.

Na Figura 16 é possível identificar que o valor do *fitness* para as configurações usando as técnicas RF, SVM e AD associadas às abordagens *online* e *batch* decai ao longo da busca, considerando o problema *Ackley*. É notório que essas configurações conseguiram aprender o comportamento da função objetivo, uma vez que aproximou-se do valor base por toda a busca. Como é sabido, nessas metologias, até a avaliação de cinquenta mil, os valores do *fitness* são calculados pela função objetivo, e, a partir disso, o *surrogate* substitui a função.

Diante disso, observa-se que antes das cinquenta mil avaliações o *fitness* já tinha chegado a um valor relativamente baixo. Porém, depois que a função objetivo foi substituída pelo *surrogate*, os valores de *fitness* preditos continuaram próximos aos valores de *fitness* do algoritmo base, o que indica sucesso por parte do *surrogate*.

A partir da Figura 16 não é possível chegar a uma conclusão em específico a respeito das três técnicas de aprendizagem de máquina quando associadas à abordagem M1, uma vez que o seu valor de *fitness* inicial é baixo, comparado ao das demais configurações. Perante isso, foi gerado um novo gráfico com apenas estas três configurações, buscando melhor entender o que ocorreu durante a busca.

Na Figura 17, nota-se que as curvas para as configurações apresentam um formato de degrau. Isso indica que a partir de um certo ponto da busca o modelo não conseguiu mais

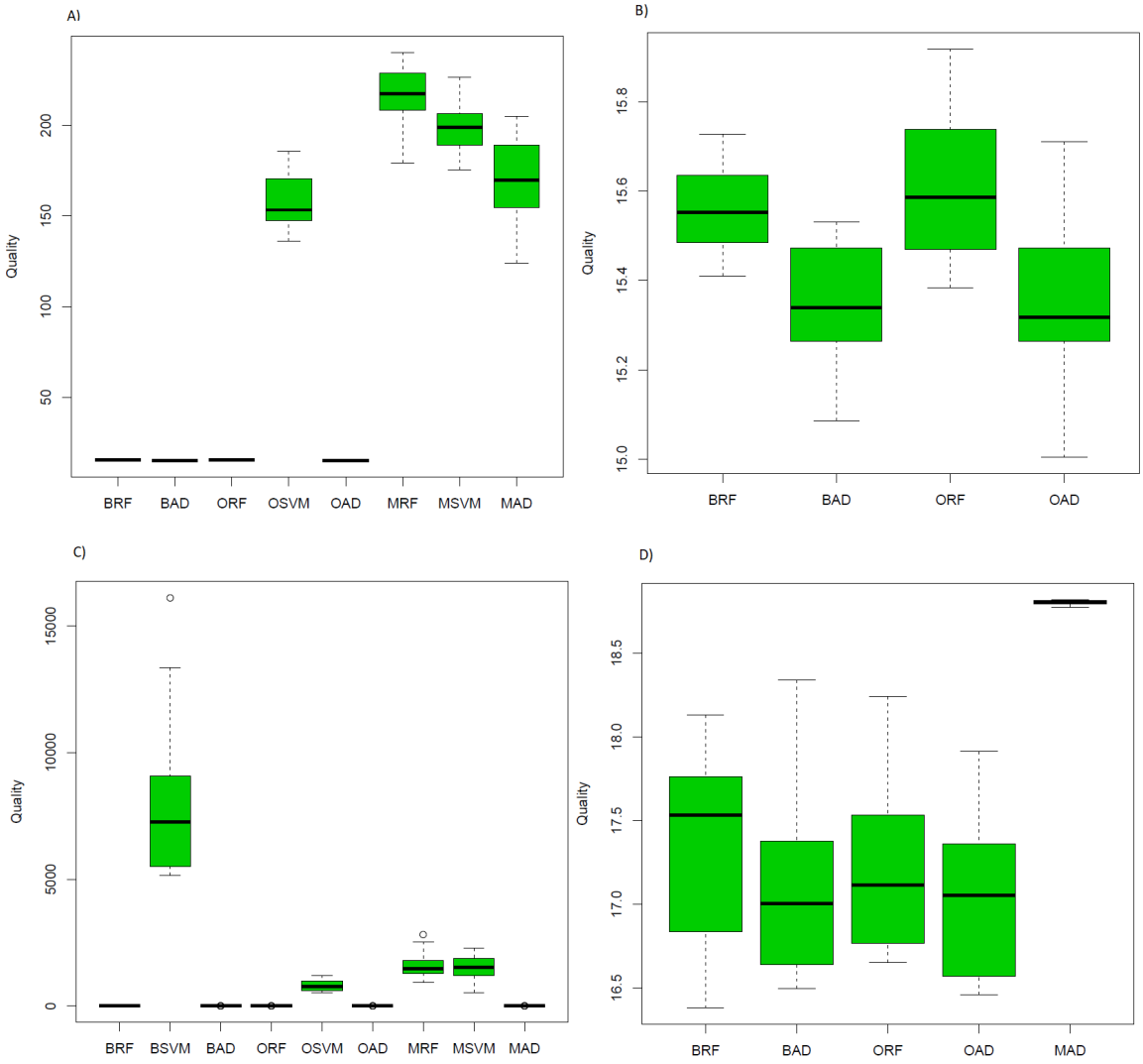

Figura 14 – Gráficos *bloxplot* para as diferentes configurações aplicadas ao problema *Rosembrock*. A) Todas as configurações *surrogate* associada ao algoritmo DE, B) melhores resultados para as configurações associada ao algoritmo DE, C) Configurações de *surrogate* associadas ao algoritmo SMPSO e D) melhores configurações de *surrogate* associadas ao algoritmo SMPSO

aprender o comportamento da função objetivo, retornando sempre a mesma faixa de valores de *fitness*, independente da solução passada como entrada.

Observando o gráfico para o problema *Ellipsoid*, ver Figura 18, nota-se que as técnicas em questão (AD, RF e SVM) conseguiram aprender o comportamento da função objetivo ao longo da busca, isso para as configurações associadas às metodologias *online* e *batch*.

Diante dos gráficos é possível identificar que, bem antes das cinquenta mil avaliações, o valor do *fitness* já estava próximo a zero. Analisando os valores dos *fitness* a partir das cinquenta mil avaliações observa-se que o *surrogate* continuou gerando valores próximos ao valor de *fitness* do algoritmo base. Ressalta-se como exceção a configuração SVM associada à abordagem *batch*,

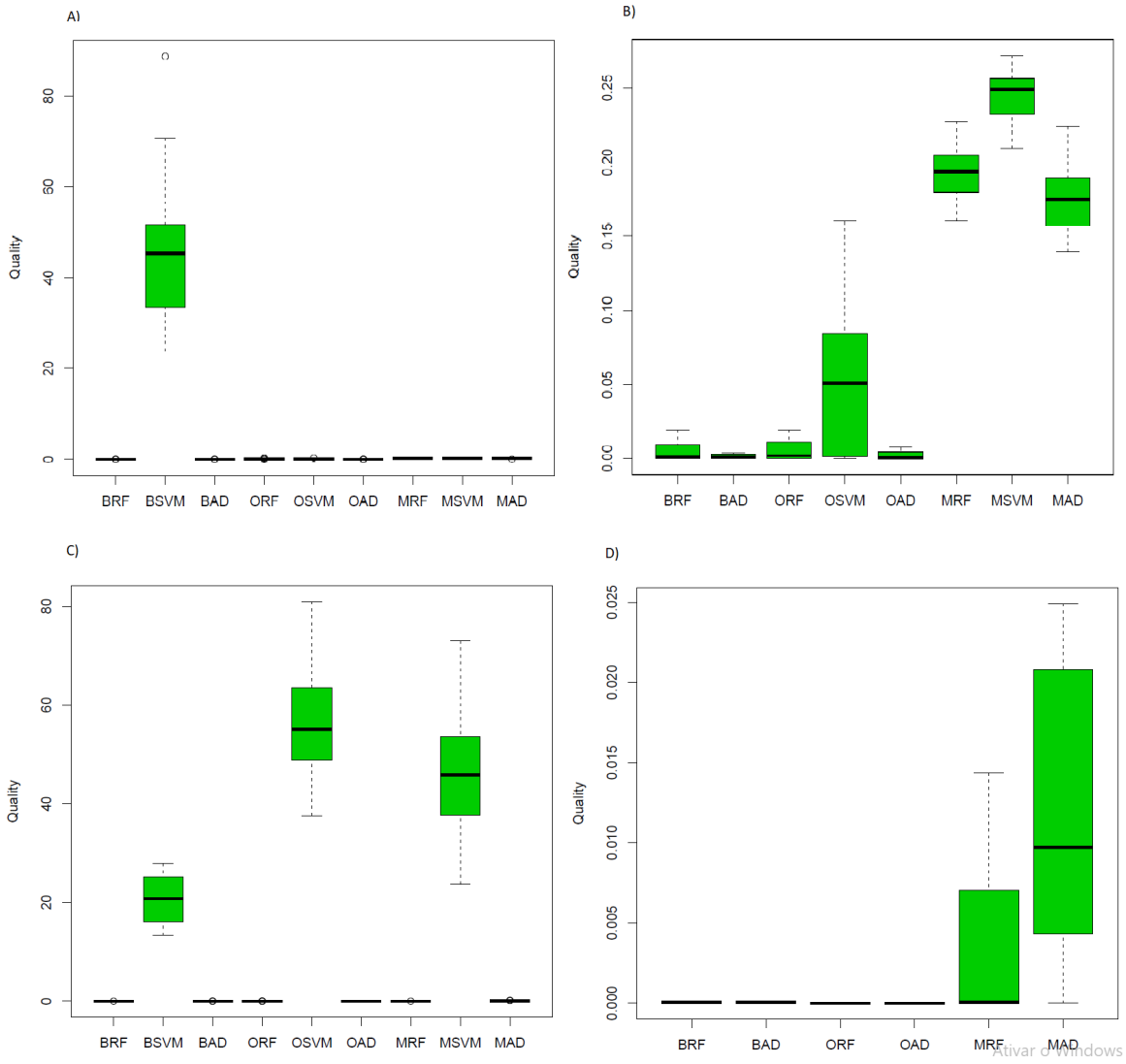

Figura 15 – Gráficos *bloxplot* para as diferentes configurações aplicadas ao problema *Griewank*. A) Todas as configurações *surrogate* associada ao algoritmo DE, B) melhores resultados para as configurações associada ao algoritmo DE, C) Configurações de *surrogate* associadas ao algoritmo SMPSO e D) melhores configurações de *surrogate* associadas ao algoritmo SMPSO

no qual os valores preditos fogem à regra, estando afastados do valor de *fitness* da função base, indicando que o treinamento não obteve o sucesso esperado.

Quando avaliado os valores de *fitness* para a abordagem M1, nota-se que eles estão proximos a zero. Porém não é possivel extrair informações a partir deles, acredita-se que por problemas de escala. Assim, foi gerado um novo gráfico apenas com as configurações que emprega a abordagem M1. O comportamento mapeado no problema anterior, *Ackley*, se manteve para as configurações usando a abordagem M1 quando aplicado ao problema *Ellipsoid*. Nota-se na Figura 19 as curvas no formato de degrau, indicando que os modelos não conseguiram aprender a função objetivo durante a busca.

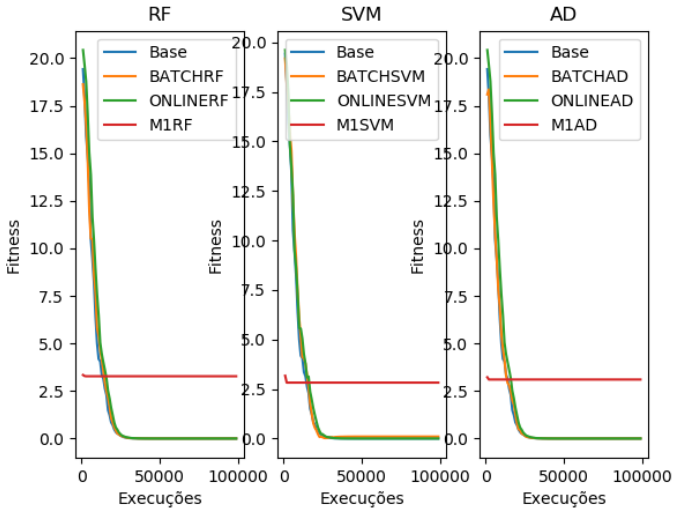

Figura 16 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Ackley* associado ao algoritmo DE

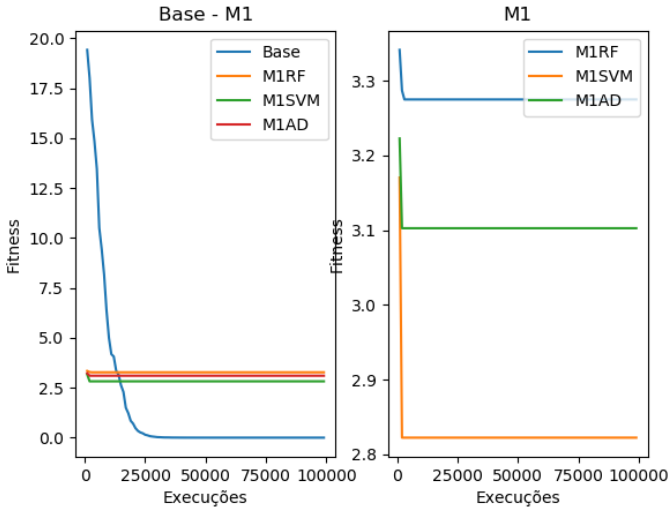

Figura 17 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Ackley* associado ao algoritmo DE para a abordagem M1

Prosseguindo com as avaliações, a Figura 20 ilustra os gráficos dos resultados para o problema *Griewank*. Para esse problema também foi possível constatar que as três técnicas obtiveram avaliações de *fitness* próximos aos *fitness* do algoritmo base. Esse problema apresentou um cenário em termos de resultado próximo ao dos dois problemas anteriores, no qual antes das cinquenta mil avaliações o valor do *fitness* já está próximo ao valor ótimo. Então, após a subtituição da função objetivo pelo *surrogate*, os valores de *fitness* preditos mantiveram uma proximidade ao *fitness* do algoritmo base.

Para o problema *Griewank* ocorreu uma pecularidade, a técnica SVM associada a abordagem *batch* obteve valores de *fitness* que foram se distanciando do valor *fitness* do algoritmo

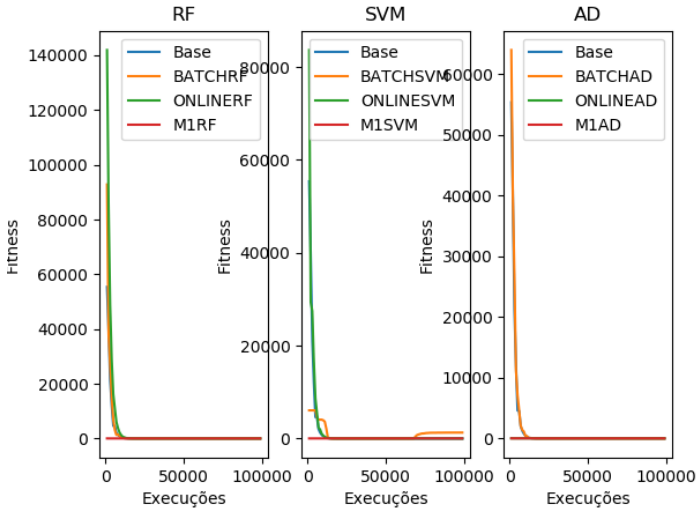

Figura 18 – Decaimento do valor do fitness no decorrer da busca para o problema *Ellipsoid* associado ao algoritmo DE

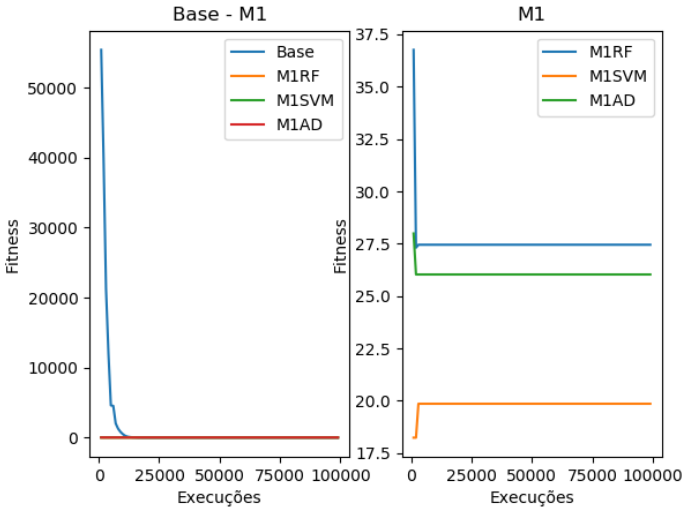

Figura 19 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Ellipsoid* associado ao algoritmo DE para a abordagem M1

base quando usado o *surrogate* para avaliar as entradas. Isso indica que o modelo não conseguiu aprender a função objetivo.

Ainda no problema *Griewank*, os valores de *fitness* para a abordagem M1, estão próximos a zero. Porém não é possível tirar conclusões dos gráficos. Assim, foi gerado um novo gráfico apenas com as configurações que emprega a abordagem M1. Na Figura 21, observa-se que o comportamento apresentado nos problemas *Ackley* e *Ellipsoid* se repetiu, no qual, novamente, foram apresentadas curvas com formato de degrau, implicando que as técnicas não obtiveram êxito em aprender a função objetivo.

Agora, tomando como base o problema *Rastrigin*, foi construído os gráficos apresentados

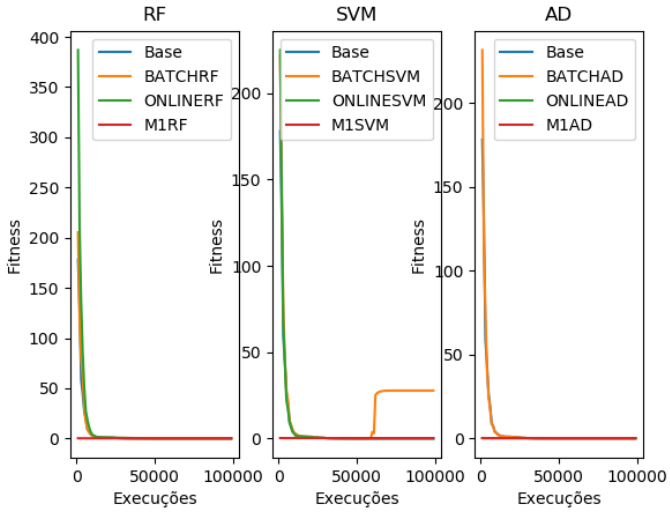

Figura 20 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Griewank* associado ao algoritmo DE

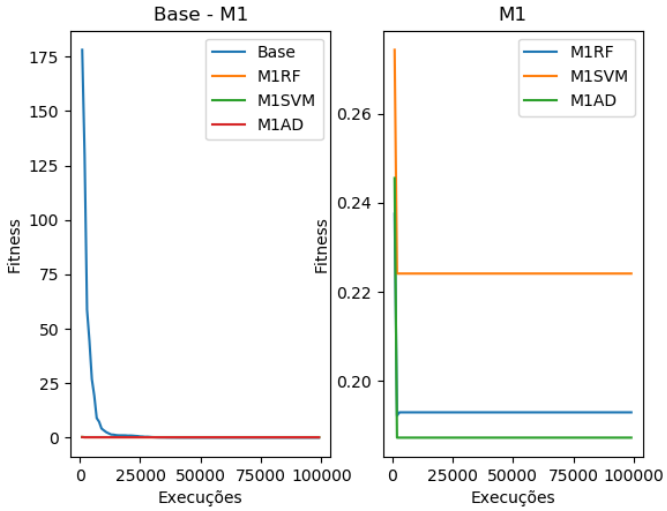

Figura 21 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Griewank* associado ao algoritmo DE para a abordagem M1

na Figura 22. Neles identificam-se que apenas as técnicas AD e SVM associadas à abordagem *batch* conseguiram aprender a função objetivo com êxito. Além disso, identifica-se que as demais técnicas, após as cinquenta mil avaliações, não conseguem predizer valores de *fitness* próximos aos valores de *fitness* do algoritmo base. Isso implica que esses modelos não conseguiram aprender a função objetivo.

Quando observado as técnicas associadas à abordagem M1, nota-se resultados confusos, nos quais inicialmente os valores dos *fitness* são baixos e depois aumentam ao decorrer da busca. Como para essa abordagem o *surrogate* é atualizado a cada ciclo de cinco interações do algoritmo evolutivo, presupoe-se que as técnicas usadas não conseguiram aprender a função objetivo do

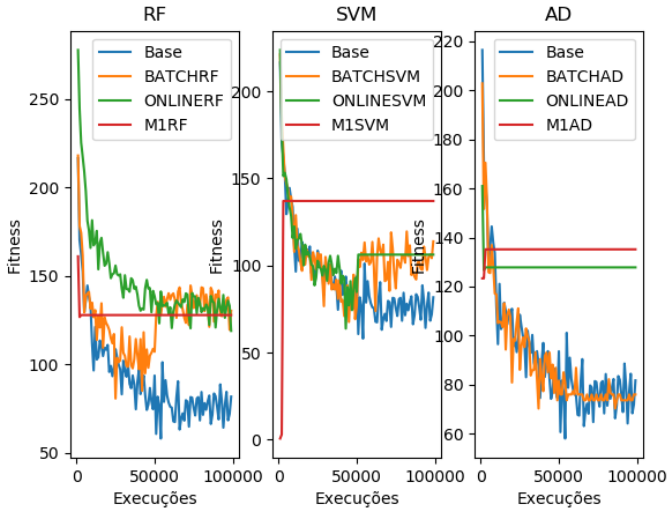

Figura 22 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Rastrigin* associado ao algoritmo DE

problema, quando associada a essa abordagem.

Por fim, na Figura 23 tem-se as ilustrações dos gráficos para o problema *Rosembrock*. Os resultados são parecidos aos dos problemas *Ackley*, *Ellipsoid* e *Griewank*. Observa-se que antes das cinquenta mil avaliações o valor do *fitness* já está próximo ao valor ótimo. Após a substituição da função objetivo pelo *surrogate*, os valores de *fitness* preditos continuaram próximos ao valor de *fitness* do algoritmo base (função objetivo real).

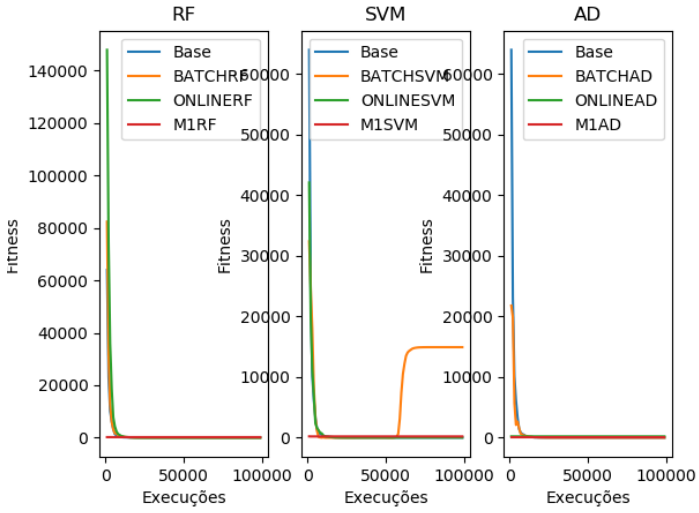

Figura 23 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Rosenbrock* associado ao algoritmo DE

Porém, para o problema *Rosembrock* tem-se como exceção a técnica SVM associada à abordagem *batch*. Para essa configuração, ao substituir a função objetivo pelo *surrogate*, os

valores de *fitness* subiram, afastando-se dos valores de *fitness* da função objetivo real. Isso, mais uma vez, implica que o modelo não conseguiu aprender a função objetivo.

Agora analisando os valores de *fitness* para as técnicas associadas a abordagem M1, observa-se que eles estão próximos a zeros, porém não é possível tirar conclusões a respeito desses resultados. Assim, foi gerado um novo gráfico para essas configurações, visando melhor entender os resultados associados a abordagem M1. Na Figura 24, é visível que para essas configurações são apresentadas curvas em formato de degrau, resultado esse parecido como os demais problemas. Diante disso, acredita-se que as técnicas não conseguiram aprender o comportamento da função objetivo do problema.

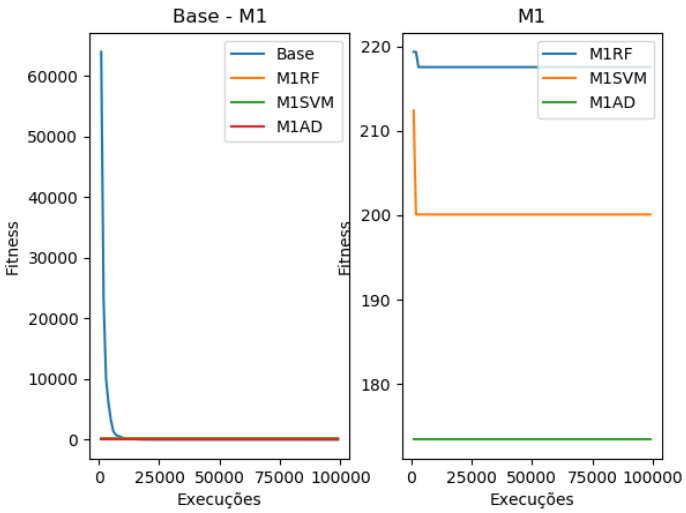

Figura 24 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Rosenbrock* associado ao algoritmo DE para a abordagem M1

Agora serão apresentados e analisados os gráficos referentes as configurações associadas ao algoritmo evolutivo SMPSO nas Figuras 25 à 29. Na primeira Figura, 25, nota-se que as técnicas de aprendizagem de máquina empregadas obtiveram bom resultados, exceto as técnicas SVM e AD associada à abordagem M1. A configuração M1SVM não conseguiu aprender o comportamento da função objetivo, enquanto a configuração M1AD, levou um maior número de avaliações da função objetivo para chegar próximo ao valor ótimo.

Nas demais configurações, as técnicas atingiram o valor ótimo antes do total de cinquenta mil avaliações. Porém, ao substituir a função objetivo pelo *surrogate* treinado, os valores preditos permaneceram próximos aos valores da função objetivo original, usada no algoritmo base. Isso evidencia que a maioria das configurações conseguiram aprender a função objetivo durante o processo da busca.

Analisando os resultados de todas as configurações para o problema *Ellipsoid* tem-se a percepção que os resultados foram bons, indicando que as técnicas conseguiram aprender a função objetivo. Porém as configurações OnlineAD, M1AD e M1RF precisaram de um número

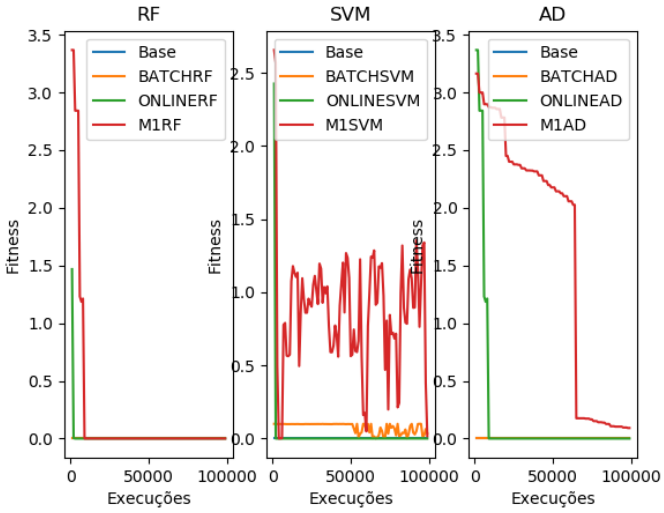

Figura 25 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Ackley* associado ao algoritmo SMPSO

maior de avaliações para chegar próximo ao valor ótimo. As demais configurações, com um baixo número de avaliações, já estavam próximos ao valor ótimo. Ao substituir a função objetivo pelo *surrogate* os valores preditos permaneceram próximos ao valor do *fitness* da função objetivo base.

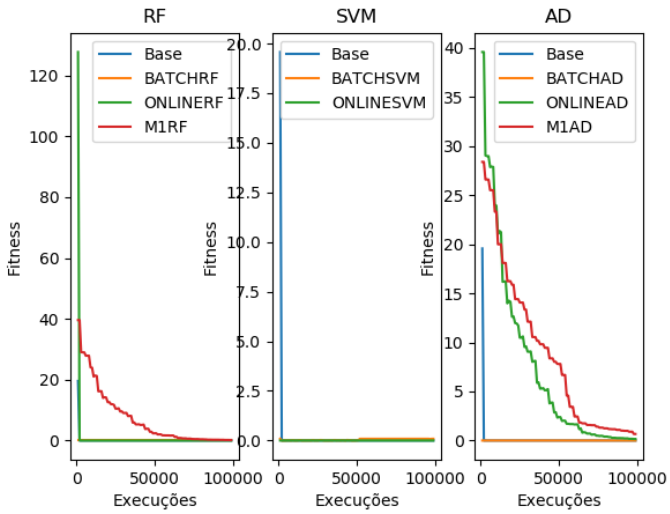

Figura 26 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Ellipsoid* associado ao algoritmo SMPSO

O mesmo comportamento do problema *Ellipsoid* não ocorreu para o problema *Griewank*. Para este, as técnicas conseguiram aprender o comportamento da função objetivo, com exceção da configuração M1SVM. Os resultados dessa configuração ilustram uma oscilação constante nos valores, indicando que a técnica não obteve êxito em aprender a função objetivo. As demais configurações conseguiram aprender a função objetivo antes das cinquenta mil avaliações, com

exceção da configuração M1AD que precisou de um número maior de avaliações para aprender a função objetivo e se aproximar do valor ótimo.

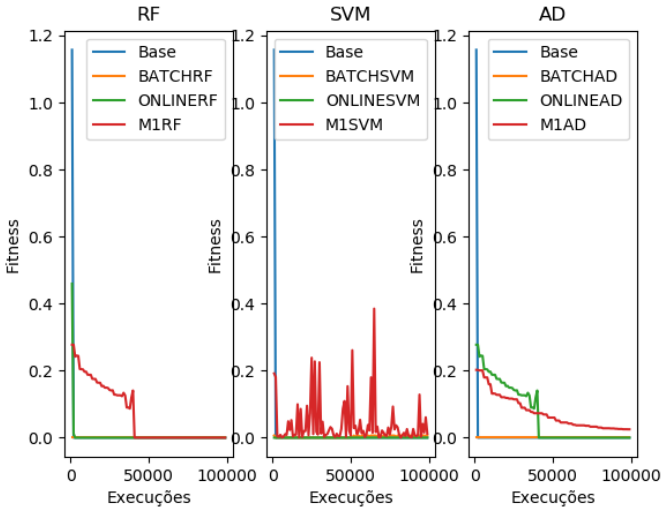

Figura 27 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Griewank* associado ao algoritmo SMPSO

No problema *Griewank* as configurações que conseguiram chegar a um valor ótimo, antes das cinquenta mil avaliações, demostram conseguir prever valores de *fitness* próximos aos *fitness* da função objetivo original, quando usado o *surrogate* para essa função, indicando assim que as técnicas conseguiram aprender a função objetivo.

Já para o problema *Rastrigin*, as técnicas apresentaram bons resultados, exceto a configuração M1SVM (esta apresentou resultados confusos e estavam atrapalhando a visualização das demais configurações, então o resultado dela foi retirado). Como observa-se, todas as configurações chegaram próximos ao valor ótimo antes das cinquenta mil avaliações, com isso, as técnicas obtiveram bons resultados quando usadas como *surrogate* substituindo a função objetivo. Os bons resultados são explicados pelo fato dos modelos serem treinados com uma gama de soluções que podem ser consideradas soluções ótimas, assim o modelo consegue prever os valores dos *fitness* fidedignamente.

Para o problema *Rosembrock* nota-se que a função objetivo atinge valores ótimos apenas próximo às cem mil avaliações da função objetivo. Diante disso, nota-se que a grande maioria das configurações apresentaram resultados atípicos. Diante dos resultados totalmente confusos apresentados pelas configurações M1RF, BatchSVM, M1SVM, OnlineAD e M1AD, as suas curvas foram retiradas dos gráficos para não atrapalhar a visualização dos resultados das demais. Presume-se que essas configurações não conseguiram aprender a função objetivo, uma vez que as curvas obtidas não seguiam um padrão de comportamento como o apresentado pela função objetivo base.

Nos resultados obtidos para o problema *Rosembrock*, nenhuma das configurações quando

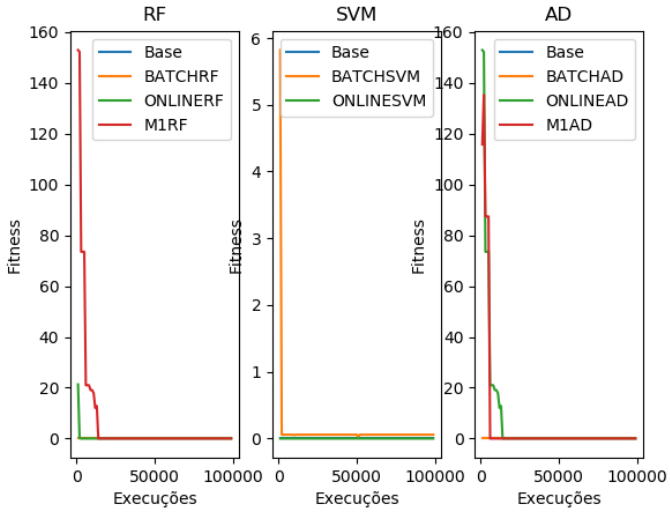

Figura 28 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Rastrigin* associado ao algoritmo SMPSO

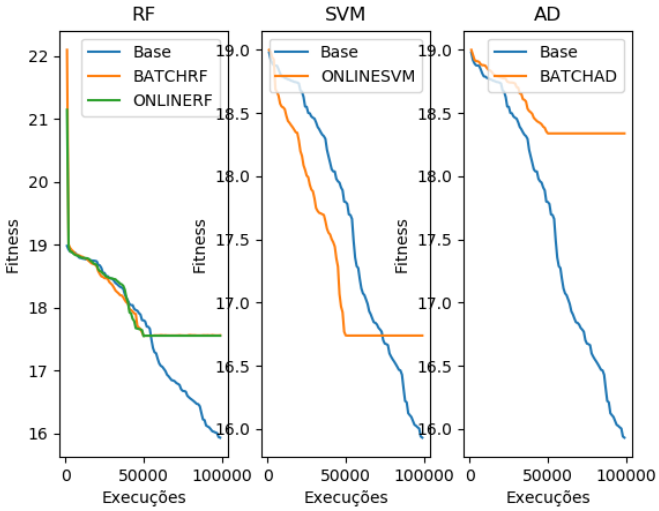

Figura 29 – Decaimento do valor do *fitness* no decorrer da busca para o problema *Rosembrock* associado ao algoritmo SMPSO

usado o modelo treinado como *surrogate* conseguiu prever valores de *fitness* próximos aos valores de *fitness* da função objetivo original. Isso implica que, anteriormente às cinquenta mil avaliações, as soluções usadas para treinar os modelos eram soluções consideradas ruins, e assim, como durante o treinamento do modelo não foram disponibilizadas soluções consideradas ótimas, os modelos não conseguiram generalizar e aprender o comportamento da função objetivo.

4.2 Experimentos e resultados para problemas multiobjetivos

Essa seção apresenta as configurações dos algoritmos empregadas nos experimentos para problemas com muitos objetivos, assim como a discussão dos resultados obtidos.

4.2.1 Planejamento dos experimentos

Seleção de Contexto: o experimento foi *in vitro* e usou problemas de otimização multiobjetivos e com muitosobjetivos simulados, de acordo com sua modelagem matemática, usando as linguagens de programação Python e Java. Foram usadas duas classes de problemas, DTLZ e WFG, sendo que a classe DTLZ possui sete diferentes problemas, DTLZ1-DTLZ7 e a classe de problemas WFG dispõe de nove problemas, WFG1-WFG9. Além disso, serão utilizados dois algoritmos de aprendizagem de máquina na forma de *surrogate* (Árvores de Decisão e *Random Forest*). Essas técnicas de aprendizagem de máquina foram empregadas na forma de *surrogate* para o algoritmo evolutivo NSGA-II empregando as abordagens dos algoritmos 1 e 2 e SMPSO quando usado a abordagem do algoritmo 3.

Variáveis Dependentes: Hiperparâmetros dos algoritmos de aprendizagem, IGD.

Variáveis Independentes: Problemas de otimização usados, DTLZ e WFG, variáveis de decisão dos problemas (apresentados na Tabela 10) e os próprios algoritmos (Árvore de Decisão e *Random Forest*).

O IGD é conhecido como Distância Geracional Invertida (*Inverted Generational Distance*) (COELLO et al., 2007b). Trata-se de uma variação da métrica conhecida como distancia geracional. O cálculo do IGD envolve o conjunto de referência de pontos pertencentes à Frenteira de Pareto global.

Projeto do Experimento: as soluções e os valores dos objetivos são armazenados de forma diferente, de acordo com a abordagem de *surrogate* usada. Na abordagem do algoritmo 1, ao atualizar a população do algoritmo evolutivo, as soluções são usadas para treinar o *surrogate*. Já na abordagem do algoritmo 2, as soluções são armazenadas em um arquivo, até atingir 20% do total de avaliações da função objetivo, e então o algoritmo de aprendizagem de máquina é treinado. Para os experimentos foram adotadas dez mil avaliações das funções objetivos como critério de parada, visto que problemas completos tendem a possuir funções objetivos com custos computacionais elevados. Em todas as abordagens (algoritmos 1, 2 e 3), assume-se que o *surrogate* está treinado ao atingir duas mil avaliações da função objetivo (20% do total de avaliações da função objetivo), então o AE deixa de usá-las, passando a usar o *surrogate* para avaliar as soluções.

Os números de avaliaçãoes da função objetivo usados como critério de parada e também para considerar o *surrogate* treinado é inferior quando comparado aos experimentos com problemas mono-objetivos. Isso se deve ao fato de problemas com muito objetivos possuir

Tabela 9 – Configurações *Surrogate* para problemas multiobjetivos

Configuração	Codinome
NSGA-II + SOmyo + AD	NSGAIISOmyoAD
NSGA-II + SBmyo + AD	NSGAIISBmyoAD
SMPSO + SOne + AD	SMPSOSOneAD
NSGA-II + SOmyo + RF	NSGAIISOmyoRF
NSGA-II + SBmyo + RF	NSGAIISBmyoRF
SMPSO + SOne + RF	SMPSOSOneRF
MOEADD	MOEADD
NSGA-II	NSGA-II

funções objetivos de alta complexidade, o que eleva o custo para avaliação de cada objetivo. Assim, normalmente na literatura é usado um número menor de soluções avaliadas.

Instrumentação: os algoritmos utilizados foram obtidos a partir de bibliotecas provenientes da linguagem de programação Python na versão 3.5.1 (PEDREGOSA et al., 2011) e jMetal (DURILLO; NEBRO, 2011). A execução dos algoritmos foi feita por meio de um computador Dell, com 8Gb de memória RAM, processador Intel i5-3470S (2.9GHz) executando em um sistema operacional sistema operacional GNU/Linux, distribuição Ubuntu 12.04 LTS.

4.2.2 Preparação

Os algoritmos evolutivos avaliam grandes volumes de soluções para encontrar as que melhor resolve um dado problema. Nesses experimentos foram usadas duas classes de problemas multiobjetivos: DTLZ e WFG. Para cada uma dessas classes foram executados dois diferentes experimentos, levando-se em consideração o número de objetivos (três e dez objetivos).

Três algoritmos meta-heurísticos, NSGA-II, SMPSO e MOEADD, foram aplicados para resolver os problemas *benchmark*. Junto aos dois primeiros foram empregados os modelos de aprendizagem de máquinas árvore de decisão e *Random Forest* como *surrogate*. O algoritmo MOEADD foi usado como algoritmo do estado da arte, com a finalidade de comparar os resultados finais. Foram criados IDs para identificar as diferentes composições de algoritmos e *surrogate* empregados. Os IDs obedecem à seguinte ordem: nome do algoritmo + abordagem de treinamento + algoritmo de aprendizado de máquina. Os diferentes IDs para as configurações implementadas estão organizados na Tabela 9. Como critério de parada, cada configuração de algoritmo, apresentada na Tabela 9, foi executada até dez mil avaliações de funções objetivos fossem alcançadas. Além disso, as versões originais dos algoritmos NSGA-II e MOEADD foram executadas sem interferência das abordagens *surrogate*.

Para cada algoritmo foi empregado os hiperparâmetros dispostos na tabela 10. Para os algoritmos evolutivos, SMPSO, NSGA-II e MOEADD foi usado os parâmetros padrões dispostos no framework jMetal.

Para os algoritmos de aprendizagem de máquina foram executados alguns exemplos

Tabela 10 – Parâmetros dos algoritmos

Algoritmo	**Parêmetros**
SMPSO	*population size* = 100, *mutation probability* = 0.05 , *mutation distribution* = 20
NSGA-II	*crossoverProbability* = 0.9, *crossoverDistributionIndex* = 20.0, *mutationProbability* = 1.0 / *number of variables*
MOEADD	*crossoverProbability* = 1.0, *crossoverDistributionIndex* = 30.0, *mutationProbability* = 1.0 / *number of variables mutationDistributionIndex* = 20.0
Random Forest	*number of estimators* = 200, *minimum samples split* = 2 , *random state* = 0, *criterion of error* = mse
Decision Tree	*maximum depth* = 500, *minimum samples split* = 2, *random state* = 0, *criterion of error* = mse

variando os principais parâmetros. Para o *Random Forest* variou-se o *number of estimators* (100, 200, 300, 400, 500) e para o árvore de decisão variou-se o *maximum depth* (200, 300, 400, 500, 1000). A partir dos exemplos executados obteve-se os conjuntos de hiperparâmetros dispostos na tabela 4 como os melhores resultados.

Os problemas *benchmark* DTLZ e WFG possuem por característica a capacidade de serem escaláveis para diferentes números de objetivos (m) e variáveis de decisão (n). Nesse contexto existe uma outra variável essencial (k), a qual define a complexidade da busca. Nos problemas DTLZ variável k é definida pela seguinte relação $k = n - m + 1$. Para todos os problemas DTLZ k foi fixada em 10. Já a classe de problemas WFG possui uma outra variável (l), relacionada à distância. Com isso o número de variáveis de decisão é definido pela relação $n = k + l + 1$. Para os prroblemas DTLZ, assumiu-se o valor de l fixo em 10 e k = 4 e 9 quando m = 3 e 10 respectivamente. Na Tabela 11 são ilustrados os valores adotados para o números de objetivos e variáveis de decisções usados nos experimentos.

Tabela 11 – Parâmetros das classes de problemas DTLZ e WFG

Problemas	**n**	**m**
DTLZ	12	3
DTlZ	19	10
WFG	15	3
WFG	20	10

As abordagens de alimentação usadas para treinar o *surrogate* possuem variáveis específicas: valor máximo de avaliação da função objetivo, *t_parada* e valor máximo de avaliação da função objetivo para treino do modelo de aprendizagem, *t_max*. Em todas as abordagens dos algoritmos 1, 2 e 3 foi usado $t\_max = 2000$ e $t\_parada = 10000$.

4.2.3 Medida de qualidade

Conforme apresentado inicialmente, as perguntas de pesquisa para essa classe de problemas serão respondidas usando como métrica de qualidade o valor da função objetivo, alcançada pelo algoritmo. Cada configuração na Tabela 9 foi executada vinte vezes, na qual o IGD de cada execução foi armazenado. As médias e desvios padrão desses vinte melhores valores de aptidão foram analisados.

Além disso, para confirmar ou refutar as hipóteses (hipótese 2 e 3), foi aplicado o teste estatístico de *Wilcoxon* (DERRAC et al., 2011). Com isso, foi possível identificar as configurações cujos resultados não apresentavam similaridades e, portanto, indicar aquelas que obtiveram os melhores resultados. O critério utilizado para escolher os melhores resultados foi o valor p_value. Para configurações com valor de p_value menor que 0,05, é possível inferir estatisticamente a diferença entre elas.

Com base nos testes estatísticos aplicados sobre os resultados e usando os testes de hipóteses:

Teste de Hipótese 2

H0: Os diferentes modelos de alimentação do *surrogate* proporcionam resultados iguais quando aplicados em algoritmos assitidos por *surrogate*;

H1: Os diferentes modelos de alimentação do *surrogate* proporcionam resultados diferentes quando aplicado em algoritmos assitidos por *surrogate*;

Teste de Hipótese 3

H0: Os algoritmos evolutivos assistidos por *surrogate* possuem resultados iguais aos algoritmos evolutivos clássicos;

H1: Os algoritmos evolutivos assistidos por *surrogate* não possuem resultados iguais aos algoritmos evolutivos clássicos;

Estima-se responder as seguintes perguntas de pesquisa.

Algoritmos Evolutivos assistidos por surrogate produzem resultados melhores ou iguais aos algoritmos evolutivos clássicos quando aplicados ao contexto de problemas com muitos objetivos?

O modelo de alimentação do surrogate influencia no desempenho do algoritmo evolutivo assistido por surrogate?

A próxima seção apresenta os resultados e a discussão. As tabelas 12, 13 14 e 15, apresentam os valores das métricas de qualidade. Os resultados destacados em negrito são aqueles em que o teste de *Wilcoxon* não computou diferenças estatísticas, ou seja, estão estatisticamente vinculados.

4.2.4 Resultados e Discussão

Essa seção dispõe dos resultados para os experimentos multiobjetivos, que encontram-se dispostos na seguinte ordem: primeiro são ilustradas as médias (do IGD) das vinte execuções e seus respectivos desvios padrões para a classe de problemas DTLZ, para três e dez objetivos (Tabelas 12 e 13, respectivamente); em seguida são abordados os resultados médios (do IGD) das vinte execuções para a classe de problema WFG, também considerando-se três e dez objetivos (Tabelas 14 e 15).

A Tabela 12 dispõe dos resultados para a classe de problema DTLZ, considerando-se três objetivos. São apresentados as médias e desvios padrões dos IGD para as configurações pertencentes a tabela 9. Todas as configurações são confrontadas buscando identificar a que obteve melhor resultado para cada problema da classe DTLZ. Foi aplicado sobre o conjunto de resultados o teste estatístico de *Wilcoxon* na versão todos contra todos e para as configurações com valor de p_value menor que 0,05, foi possível inferir diferença estatística. A partir dela observa-se que as abordagens que empregam um modelo de aprendizagem para cada objetivo (algoritmos 1 e 2) obtiveram bons resultados.

Dentre essas, destacam-se as configurações NSGAIISOmyoAD para o problema DTLZ2, a qual obteve o melhor índice IGD para esse problema, empatada com o algoritmo original *NSGA-II* e a configuração NSGAIISOmyoRF para o problema DTLZ5 obteve o melhor IGD, empatada com o algoritmo do estado da arte *MOEADD*.

Analisando de forma geral os resultados para as configurações que envolvem os algoritmos 1 e 2 (NSGAIISOmyoAD, NSGAIISBmyoAD, NSGAIISOmyoRF, NSGAIISBmyoRF) os resultados obtidos foram satisfatórios, uma vez que os IGDs estão próximos aos IGDs obtidos pelo algoritmo original *NSGA-II* e até mesmo do algoritmo do estado da arte *MOEADD*.

Um fator importante a ser constatado é que a abordagem 3 usa 20% do total de avaliações da função objetivos para treinar o *surrogate*, após isso é usado o modelo já treinado, e obteve resultados semelhantes aos algoritmos de otimização originais. Esse fator tem uma relevância uma vez que será reduzido de forma significativa o número de chamada da função objetivo, assim reduzindo o custo computacional de toda a busca.

Nesse contexto existem alguns pontos cujos IGDs para algumas configurações encontram-se totalmente divergentes quando comparado com IGD obtido pelo algoritmo original *NSGA-II*. São eles NSGAIISBmyoAD e NSGAIISOmyoRF para o problema DTLZ3 e NSGAIISOmyoAD para o problema DTLZ6.

Para esse cenário o algoritmo original *NSGA-II* obteve resultados melhores (com uma diferença consistente) que os resultados obtidos pelas configurações usando *surrogate*. Aqui ressalta-se que nos problemas DTLZ3 e DTLZ6 o intervalo dos valores dos objetivos são maiores e isso pode ser considerado um problema. Uma vez que, as soluções iniciais, que ainda estão longe da fronteira, possuem valores altos. Assim, o *surrogate* (treinado com as soluções iniciais)

aprende a função objetivo numa faixa de valores diferentes do que é esperado ao final. Assim, as configurações usando *surrogate* tem uma dificuldade em convergir. Esse mesmo cenário pode ocorrer no problema DTLZ1.

Analisando os resultados obtidos para as configurações que envolvem o algoritmo 3 (SMPSOSOneAD e SMPSOSOneRF) constata-se que estas obtiveram um conjunto de resultados consistentes e satisfatórios. Isso fica evidente pelo fato dessas configurações obterem os melhores conjuntos de IGDs para três diferentes problemas, DTLZ1, DTLZ3 empatado com *MOEADD* e DTLZ6 empatado com *MOEADD* e *NSGA-II*.

Observando-se os resultados gerais, as outras abrdagens usando *surrogate* (algoritmos 1 e 2) obtiveram resultados inferiores em relação a abordagem do algoritmo 3 para os problemas DTLZ1, DTLZ3 e DTLZ6. Acredita-se que isso tem uma relação ao algoritmo empregado pela abordagem 3, SMPSO.

Uma explicação para o ocorrido é o fato do algortimo PSO geralmente convergir mais rápido. Isso pode fazer que ele caia em ótimos locais. Nesse caso ele conseguiu bons valores de objetivo para esses problemas ainda nas soluções iniciais, o que fez com que o *surrogate* obtivesse melhores resultados.

Ainda para essas configurações observa-se que os problemas nos quais não obteve-se os melhores índices IGD, os valores encontram-se próximos aos melhores IGD obtidos para cada problema em específico.

Tabela 12 – Resultados médios das diferentes configurações de *surrogate* para a classe de problemas DTLZ com 3 Objetivos

	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
NSGAIISOmyoAD	0.13175 ± 0.1441	**0.000129 ± 2.63e-05**	0.0266 ± 0.0065	0.00422 ± 0.00266	0.00217 ± 0.00032	0.75713 ± 0.71693	0.02984 ± 0.0030
NSGAIISBmyoAD	0.44419 ± 0.09584	0.001226 ± 0.000112	1.91966 ± 0.4306	0.00440 ± 0.00290	0.000125 ± 4.105e-05	0.03981 ± 0.00182	0.03026 ± 0.0112
NSGAIISOmyoRF	0.45805 ± 0.1185	0.001156 ± 0.00015	1.74752 ± 0.5371	0.004943 ± 0.003059	**9.59e-05 ± 3.95e-05**	0.03765 ± 0.00390	0.03110 ± 0.010039
NSGAIISBmyoRF	0.45418 ± 0.10679	0.001379 ± 0.00012	1.8099 ± 0.2681	0.00333 ± 0.00203	0.00012 ± 4.44e-05	0.04362 ± 0.00122	0.039194 ± 0.01521
SMPSOSOneAD	**0.00206 ± 9.087e-05**	0.00788 ± 0.00302	**0.00822 ± 0.00153**	0.00861 ± 0.000858	0.00144 ± 0.00050	**0.01163 ± 0.00558**	0.050475 ± 0.016822
SMPSOSOneRF	**0.002035 ± 0.00010**	0.00631 ± 0.00172	**0.00729 ± 0.00052**	0.00864 ± 0.00109	0.00154 ± 0.00043	**0.00935 ± 0.004329**	0.04686 ± 0.01725
MOEADD	0.02474 ± 0.0085	0.00596 ± 0.00046	**0.00068 ± 5.62e-06**	**0.000107 ± 4.5e-06**	0.0149 ± 0.00082	**0.0005976 ± 9.3e-07**	0.140604 ± 0.05392
NSGA-II	0.10865 ± 0.05042	**0.00080 ± 3.21e-05**	0.37121 ± 0.11127	0.00342 ± 0.002841	**1.88e-05 ± 6.40e-07**	**0.01932 ± 0.001541**	**0.006415 ± 0.00856**

Visto que as configurações empregadas obtiveram bons resultados quando empregadas na classe de problemas DTLZ para três objetivos, a seguir é apresentada uma análise dos resultados para as mesmas configurações (Tabela 9) aplicadas à classe de problema DTLZ com dez objetivos. Os resultados estão dispostos na Tabela 13.

Observa-se que os bons resultados obtidos pelas configurações que empregam os algoritmos 1 e 2 (NSGAIISOmyoAD, NSGAIISBmyoAD, NSGAIISOmyoRF, NSGAIISBmyoRF) permanecem quando aplicado a mesma classe de problema, porém, com um número maior de objetivos (dez objetivos).

Enquanto para os problemas com três objetivos as configurações NSGAIISBmyoAD e NSGAIISBmyoRF obtiveram índice IGD apenas próximo aos índices IGD do algoritmo original *NSGA-II* e *MOEADD*, para os problemas com dez objetivos elas começaram a figurar entre os melhores IGDs obtidos para os problemas DTLZ2, DTLZ4 e DTLZ5.

As demais configurações NSGAIISOmyoAD e NSGAIISOmyoRF também mantiveram bom desempenho, obtendo índices de IGDs dentre os melhores para os problemas DTLZ2, DTLZ4 e DTLZ5 (apenas NSGAIISOmyoRF), sendo que a configuração NSGAIISOmyoAD ainda obteve índice IGD entre os melhores para os problemas DTLZ1 e DTLZ3.

Analisando os resultado de forma geral, observa-se que, para a maioria dos problemas, as configurações que empregam os algoritmos 1 e 2 obtiveram índices IGDs semelhantes estatisticamente ou IGDs bém próximos aos valores IGD para o algoritmo original *NSGA-II* ou algoritmo do estado da arte *MOEADD*.

Nesse contexto destacam-se alguns cenários específicos em que as configurações assistidas por *surrogate* obtiveram índices IGDs muito melhores que o índice IGD do algoritmo original *NSGA-II*, sendo elas os problemas DTLZ1 e DTLZ3 para a configuração NSGAIISOmyoAD em detrimento ao algoritmo original *NSGA-II*. Esse caso é o oposto ao ocorrido para os problemas com três objetivos e requer uma análise aprofundada para investigar os possíveis fatores para esse acontecimento.

Ao averiguar e avaliar os resutados obtidos para as configurações SMPSOSOneAD e SMPSOSOneRF observa-se que em todos os problemas, exceto o DTLZ4, elas obtiveram os valores de IGDs dentre os melhores. Sendo que, em alguns casos, tais configurações superaram o algoritmo original NSGA-II e do estado da arte MOEADD (DTLZ1, DTLZ6 e DTLZ7).

Tabela 13 – Resultados medios das diferentes configurações de *surrogate* para a classe de problemas DTLZ com 10 Objetivos

	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
NSGAIISOmyoAD	**0.004541** ± **0.000226**	**0.002727** ± **0.000315**	**0.061875** ± **0.004171**	**0.01656** ± **0.0008749**	3.076414 ± 0.62778	12.73145 ± 1.55513	0.59820 ± 0.05675
NSGAIISBmyoAD	2.89464 ± 0.50902	**0.005347** ± **0.00034**	14.16820 ± 1.60551	**0.01644** ± **0.00073**	**0.00328** ± **0.00037**	0.061088 ± 0.005932	0.644239 ± 0.059398
NSGAIISOmyoRF	3.52041 ± 0.445967	**0.00622** ± **0.00044**	16.66517 ± 1.59656	**0.01843** ± **0.00088**	**0.00346** ± **0.00038**	0.0549464 ± 0.003287	0.610886 ± 0.0679053
NSGAIISBmyoRF	3.34270 ± 0.47323	**0.006485** ± **0.000512**	16.73269 ± 1.16006	**0.018402** ± **0.00087**	**0.003692** ± **0.00036**	0.05663 ± 0.003768	0.642572 ± 0.061817
SMPSOSOneAD	**0.00730** ± **0.00068**	**0.00760** ± **0.001366**	**0.025649** ± **0.001086**	0.02661 ± 0.001502	**0.007025** ± **0.00154**	**0.01595** ± **0.005619**	**0.142983** ± **0.067263**
SMPSOSOneRF	**0.00738** ± **0.00054**	**0.007404** ± **0.001609**	**0.025475** ± **0.001188**	0.02659 ± 0.000989	**0.005776** ± **0.0020065**	**0.019620** ± **0.00890**	**0.11908** ± **0.05626**
MOEADD	0.01980 ± 0.000840	1.028180 ± 0.19597	**0.00186** ± **0.00023**	0.32315 ± 0.047625	**0.002359** ± **5.84e-05**	0.07399 ± 0.00157	3.56218 ± 0.60088
NSGA-II	5.55687 ± 0.37507	**0.00799** ± **0.00067**	14.16424 ± 2.87821	0.026259 ± 0.001115	**0.00301** ± **0.000474**	0.08392 ± 0.001689	0.33371 ± 0.08158

O uso da técnica KKTPM nas abordagens SMPSOSOneAD e SMPSOSOneRF é uma forte justificativa para esses bons resultados, uma vez que a técnica KKTPM é empregada com a finalidade de restringir as infinitas soluções ótimas da fronteira de Pareto em uma função discreta com um conjunto finito de soluções.

Daqui em diante será abordado os resultados para os experimentos usando o problema WFG, sendo incialmente explanado os resultados para os experimentos com três objetivos e em seguida os experimentos usando dez objetivos.

Ao observar os resultados obtidos para as configurações assistidas por *surrogate* usando os algoritmos 1 e 2, ver Tabela 14, nota-se que, a exemplo da classe de problemas DTLZ, os resultados obtidos são bons.

Em destaque tem-se as configurações NSGAIISOmyoAD, com os IGDs para os problemas WFG1 e WFG7 dentre os melhores IGDs, NSGAIISBmyoAD, a qual obteve os melhores IGDs para os problemas WFG2, WFG3 e WFG4, NSGAIISOmyoRF, que obteve IGDs considerados os melhores para os problemas WFG2, WFG5, WFG7, WFG8, WFG9 e NSGAIISBmyoRF, com os melhores IGDs para os problemas WFG2, WFG4, WFG5, WFG7 e WFG9.

Ressalta-se que todas essas configurações obtiveram os seus melhores resultados empatadas estatisticamente com outras configurações, dentre as quais a configuração usando o algoritmo original *NSGA-II*. Além disso, quando não obtiveram índices IGDs dentre os melhores, os valores encontraram-se próximos aos valores de IGDs obtidos pelos algoritmos original e do estado da arte, *NSGA-II* e *MOEADD*.

Todas essas configurações empregam um modelo de aprendizagem para cada função objetivo do problema. Ao analisar os resultados para as configurações SMPSOSOneAD e SMPSOSOneRF, as quais empregam um único modelo de aprendizagem independente do número de funções objetivos, nota-se que obteve-se apenas o IGD para o probema WFG1.

Ao observar os valores de IGD obtidos para essas configurações, nota-se que eles estão próximos aos valores de IGD obtidos pelas configurações do algoritmo *NSGA-II* e *MOEADD*.

Tabela 14 – Resultados medios das diferentes configurações de *surrogate* para a classe de problemas WFG com 3 Objetivos

	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
NSGAIISOmyoAD	*0.01717* ± *0.00284*	0.05321 ± 0.00209	0.01474 ± 0.00275	0.01425 ± 0.00313	0.01647 ± 0.00764	0.01427 ± 0.0062	*0.00426* ± *0.00146*	0.01061 ± 0.00195	0.01944 ± 0.01183
NSGAIISBmyoAD	0.05211 ± 0.001117	**0.00913** ± **0.00187**	**0.00234** ± **0.00059**	0.00746 ± 0.00216	0.00673 ± 0.00137	0.00479 ± 0.0007	0.01324 ± 0.00373	**0.00442** ± **0.00029**	0.02582 ± 0.00875
NSGAIISOmyoRF	0.05215 ± 0.00113	**0.00820** ± **0.001598**	0.00307 ± 0.00032	0.00493 ± 0.00087	**0.00462** ± **0.00051**	0.00520 ± 0.00070	**0.00462** ± **0.00041**	**0.00508** ± **0.00035**	**0.00476** ± **0.00071**
NSGAIISBmyoRF	0.05225 ± 0.00105	**0.00884** ± **0.00106**	0.00376 ± 0.00049	**0.00349** ± **0.00023**	**0.00399** ± **0.00021**	0.00524 ± 0.00037	**0.00464** ± **0.00053**	0.00639 ± 0.00028	**0.00492** ± **0.00067**
SMPSOSOneAD	**0.03566** ± **0.001076**	0.01552 ± 0.00266	0.01176 ± 0.00312	0.00798 ± 0.00120	0.01986 ± 0.00617	0.01672 ± 0.00516	0.01504 ± 0.00525	0.02320 ± 0.00769	0.01383 ± 0.00358
SMPSOSOneRF	**0.03504** ± **0.00052**	0.01426 ± 0.00151	0.01135 ± 0.0033	0.00775 ± 0.00087	0.02047 ± 0.0059	0.01825 ± 0.00765	0.01314 ± 0.00526	0.01923 ± 0.00647	0.01268 ± 0.00428
MOEADD	**0.02022** ± **0.00604**	**0.00311** ± **0.00012**	0.04328 ± 0.001140	**0.00293** ± **0.00030**	0.00302 ± 0.00030	**0.00276** ± **4.45e-05**	**0.00288** ± **0.00019**	**0.00293** ± **4.84e-05**	**0.00406** ± **7.11e-05**
NSGA-II	**0.00474** ± **0.00071**	**0.00281** ± **0.00014**	**0.00318** ± **0.00015**	**0.00346** ± **0.00016**	**0.00148** ± **0.00027**	**0.002940** ± **0.00010**	**0.04001** ± **0.00182**	**0.00465** ± **8.90e-05**	**0.00336** ± **0.00044**

De forma geral observa-se que, para os problemas WFG, as configurações que empregam modelos de aprendizagem individuais para cada função objetivo, obteve os melhores resultados quando comparado com as abordagens que usam um único modelo de aprendizagem, oposto do ocorrido para a classe de problemas DTLZ, fato discutido com maior enfoque na Seção 4.2.5.

A seguir, na Tabela 15 são apresentados os resultados para os experimentos usando a classe de problema WFG com dez funções objetivos. Ao analisar os IGDs obtidos, observa-se que nesse contexto as abordagens que empregam modelos de aprendizagem individuais para cada função objetivo se sobressaiu em relação às configurações que empregam um único modelo de aprendizagem.

Em suma, as configurações NSGAIISOmyoAD, NSGAIISBmyoAD, NSGAIISOmyoRF, NSGAIISBmyoRF obtiveram bons resultados, apresentando os melhores índices IGD para problemas como WFG3, WFG5, WFG6, WFG8 e WFG9, enquanto as configurações SMPSOSOneAD e SMPSOSOneRF obteve o IGD para o problema WFG3 dentre os melhores IGD computados.

No geral quando observa-se os valores de IGDs obtidos por todas as configurações assistidas por *surrogate* (NSGAIISOmyoAD, NSGAIISBmyoAD, NSGAIISOmyoRF, NSGAIISBmyoRF, SMPSOSOneAD e SMPSOSOneRF), nota-se que estes quando não estão empatados estatisticamente com os IGDs do algoritmo original *NSGA-II* ou do algoritmo do estado da arte, *MOEADD*.

Entretanto, os resultados obtidos pelas configurações SMPSOSOneAD e SMPSOSOneRF para a classe de problema WFG requer uma análise minunciosa, uma vez que para a classe de problema DTLZ essas configurações figurou entre os melhores resultados e para a classe de problema WFG o seu desempenho foi abaixo do esperado.

Tabela 15 – Resultados medios das diferentes configurações de *surrogate* para a classe de problemas WFG com 10 Objetivos

	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
NSGAIISOmyoAD	0.12663 ± 0.00146	0.11571 ± 0.00248	0.11659 ± 0.00241	0.05228 ± 0.00201	**0.10270 ± 0.00243**	**0.10787 ± 0.00143**	0.12063 ± 0.001775	**0.11631 ± 0.00122**	**0.04037 ± 0.00155**
NSGAIISBmyoAD	0.10734 ± 0.0010	0.05137 ± 0.00180	**0.04266 ± 0.00224**	0.10050 ± 0.00159	**0.11646 ± 0.00125**	**0.12286 ± 0.001241**	0.11489 ± 0.00148	**0.11354 ± 0.00126**	**0.11585 ± 0.00236**
NSGAIISOmyoRF	0.10850 ± 0.00092	0.05407 ± 0.00276	**0.04110 ± 0.00241**	0.10373 ± 0.00213	**0.11612 ± 0.00097**	**0.11778 ± 0.00175**	0.11108 ± 0.00165	0.11146 ± 0.00119	**0.11985 ± 0.00212**
NSGAIISBmyoRF	0.10881 ± 0.00139	0.05328 ± 0.00168	**0.04072 ± 0.00238**	0.10445 ± 0.00185	**0.11412 ± 0.00174**	**0.11799 ± 0.00134**	0.11171 ± 0.00210	0.11019 ± 0.00095	**0.11996 ± 0.00211**
SMPSOSOneAD	0.12275 ± 0.00428	0.11183 ± 0.02436	**0.05939 ± 0.01431**	0.26185 ± 0.02345	0.32861 ± 0.07472	0.42343 ± 0.05981	0.39139 ± 0.06619	0.50301 ± 0.07305	0.25486 ± 0.06865
SMPSOSOneRF	0.11868 ± 0.00619	0.11796 ± 0.02578	**0.05015 ± 0.011705**	0.24695 ± 0.01744	0.30439 ± 0.04987	0.40948 ± 0.05457	0.35755 ± 0.080226	0.48302 ± 0.07371	0.23050 ± 0.04124
MOEADD	0.22032 ± 0.01040	**0.03595 ± 0.00146**	0.18136 ± 0.0067	0.20697 ± 0.010713	0.18968 ± 0.00929	0.20227 ± 0.00660	**0.043164 ± 0.00162**	**0.10622 ± 0.00210**	0.13311 ± 0.00275
NSGA-II	**0.04872 ± 0.00131**	0.11521 ± 0.00088	0.12153 ± 0.00088	**0.03362 ± 0.0038**	**0.10977 ± 0.00148**	**0.09641 ± 0.00108**	0.11219 ± 0.00083	0.11732 ± 0.00096	0.12264 ± 0.00100

4.2.5 Considerações Finais

As abordagens de treino de *surrogate* propostas proporcionaram resultados satisfatórios para as diferentes configurações de experimentos presentes nas Tabelas 3 e 9. Na primeira Seção de experimentos foi possível constatar que as abordagens assistidas por modelos de Aprendizagem de Máquina conseguiu aprender o comportamento da função objetivos para problemas mono-objetivos.

Nesse contexto, ressalta-se a necessidade de garantir que a base de treino do modelo de aprendizagem possua soluções de entradas diversificadas, possibilitando que o modelo aprenda de forma genérica, de modo que possa generalizar e distinguir entre as soluções boas e ruins.

Para os experimentos com problemas mono-objetivos foi possível identificar que as técnicas de aprendizagem de máquina RF e AD obtiveram os melhores resultados, estando empatadas estatisticamente na maioria das configurações empregadas.Uma possível justificativa para essas abordagens possuirem os melhores resultados são os hiperparâmetros adotados no treinamentos das tecnicas de Aprendizagem de Máquina.

Levando-se em conta esses resultados e as curvas de aprendizagem apresentadas, refuta-se H0 da hipótese 1, assumindo-se que H1 é verdadeira. Com isso, infere-se que as técnicas de aprendizagem não possuem os mesmo desempenho em termos de eficácia, e com isso as abordagens RF e AD produziram os melhores resultados quando empregadas como *surrogate* em problemas de otimização mono-objetivos.

Agora, observando-se os diferentes resultados das configurações de algoritmos (tabela 3) em relação ao modelo de alimentação do *surrogate*, nota-se que os modelos de alimentação S_Online e S_Batch se destacaram quando associados as técnicas de aprendizagem de máquina AD e RF. As configurações empregando S_Online ou S_Batch atreladas as técnicas AD ou RF produziram resultados semelhantes para grande parte das configurações de algoritmos executados. Porém, levando-se em conta que os resultados obtidos pelas configurações usando M1 e também as configurações S_Online e S_Batch associadas a técnica de aprendizagem SVM produziram resultados estatisticamente diferentes, refulta-se H0 da hipótese 2, assim, toma-se H1 da hipótese 2 como verdadeira. Com isso, infere-se que o modelo de alimentação do *surrogate* não influencia diretamente no desempenho do algoritmo evolutivo assistido por *surrogate*.

Partindo para as abordagens propostas para problemas multiobjetivos, observa-se que os *frameworks* de treinamento dos modelos, algoritmos 1, 2 e 3, associadas às técnicas de aprendizagem de máquina AD e RF, obtiveram bons resultados, em grande parte obtendo índices IGDs melhores ou iguais aos índices IDGs dos algoritmos *NSGA-II* e *MOEADD*.

No contexto dos experimentos com problemas multiobjetivos constatou-se que em alguns casos específicos é preciso a realização de um estudo mais detalhado para identificar as possíveis causas das divergências encontradas nos resultados. São elas, a consistente divergência nos índices IGDs entre o NSGA-II e as configurações NSGAIISBmyoAD e NSGAIISOmyoRF para o

problema DTLZ3 e NSGAIISOmyoAD para o problema DTLZ6, quando abordado os problemas com três objetivos. Já para os problemas com dez objetivos existe uma divergência de IGD entre o NSGA-II e a configuração NSGAIISOmyoAD para os problema DTLZ1 e DTLZ3.

Além desses casos para a classe de problemas DTLZ, observa-se que o desempenho das abordagens SMPSOSOneAD e SMPSOSOneRF tiveram uma queda de desempenho quando comparado às classes de problemas DTLZ e WFG. Enquanto na classe de problemas DTLZ essas abordagens figuraram entre os melhores resultados, para a classe de problemas WFG esse desempenho não se repetiu.

Existem duas possíveis explicações para esse fato, a de que os problemas WFG possuem maior complexidade que a classe DTLZ e por isso as abordagens não conseguiram obter bom desempenho, e a outra está embasada no algoritmo evolutivo usado pelas abordagens (SMPSOSOneAD e SMPSOSOneRF), SMPSO. É possível que o algoritmo SMPSO apresente dificuldades para encontrar as melhores soluções para a classe de problemas WFG.

Ressalta-se que essas sugestões são inferências e não foram comprovadas a partir de experimentos e estudos com uma análise aprofundada, ficando assim como pontos em abertos a serem explorados em trabalhos futuros.

Analisando os resultados obtidos e apresentados nas tabelas 12, 13, 14 e 15) refuta-se H0 da hipótese 3, uma vez que os resultados evidenciam que nem sempre os algoritmos assistidos por *surrogate* obtiveram os melhores índices IGDs e, na grande maioria, quando as configurações de algoritmos assistidos por *surrogate* obteve os melhores resultados, estavam empatadas com algum dos algoritmos do estado da Arte.

Esse fato induz a aceitação de H1 da hipótese 3. Assim, infere-se que os algoritmos evolutivos assistidos por *surrogate* não produz resultados melhores que os algoritmos evolutivos clássicos. Observa-se que os algoritmos evolutivos assistidos por *surrogate* em geral apresentam índices IGDs semelhantes estatisticamente aos dos AE classicos, ou então, apresentam IGDs relativamente próximos.

Ainda a partir dos resultados, refuta-se H0 da hipótese 2. Assim, não existem evidências suficientes para rejeitar H1. Isso fica evidente a partir do conjunto de experimentos usando a abordagem de alimentação *S_One*. Para a classe de problemas DTLZ, as configurações usando a abordagem *S_One* figuram entre os melhores índices IGDs coletados, porém as mesmas configurações quando aplicadas a classe de problema WFG, obtiveram índices IGDs distantes dos melhores índices coletados.

Assim, é possível ressaltar que a metodologia de alimentação usada não causa influência direta no desemenho do AE assistido por *surrogate*. Existem alguns fatores que podem interferir diretamente no desempenho do AE assistido por *surrogate*, como, a capacidade de aprendizagem e generalização do modelo de aprendizagem empregado, dificuldade do problema, características do método de busca empregada pelo AE, impossibilitando que o sejam disponibilizadas soluções

heterogêneas ao modelo de aprendizado.

5
Conclusões

Os algoritmos evolutivos assistidos por *surrogate* têm ganhado uma maior atenção nos últimos anos, o que tem feito com que os pesquisadores aprimorem as abordagens de *surrogate* até então utilizadas. Até recentemente a maior parte dos *surrogates* empregados consistiam de um único *surrogate* para aprender cada função objetivo do problema. Trabalhos recentes têm evoluido esse aspecto, conseguindo desenvolver novas abordagens que empregam um único *surrogate* para aprender as *m* diferentes funções objetivos do problema.

Nesse trabalho foi empregado e avaliado novas abordagens de treinamento de *surrogate* associados a AE. Essas novas abordagens de treinamento consistem da metodologia empregada para disponibilização dos dados ao modelo de aprendizagem aplicado como *surogate*. Foram avaliadas abordagens propostas pelo autor, assim como metodologias já disponíveis na literatura. Além disso, foi analisado o desempenho de diferentes modelos de aprendizagem na forma de *surrogate*, que, até onde se tem conhecimento, ainda não foram usados para essa finalidade.

Inicialmente foram desenvolvidas abordagens de treino do *surrogate* para problemas mono-objetivos. Os experimentos para essas abordagens mostraram bons resultados, indicando que as metodologias de treino do *surrogate* usadas obtiveram bom desempenho, gerando modelos de aprendizagem que imitam o comportamento da função objetivo do problema.

Como ressalva para essa etapa inicial dos experimentos salienta-se a necessidade de controlar as soluções usadas na fase de treinamento do modelo de aprendizagem, ou seja, se faz necessário o uso de soluções consideradas boas e ruins visando uma maior capacidade de generalização do modelo treinado.

Considerando as abordagens que englobam os problemas com muitos objetivos, empregando classificadores individuais para cada função objetivo do problema, assim como um único classificador para *m* diferentes funções objetivos, produziu bons índices IGDs. Os resultados obtidos para ambas as abordagens são promissores e satisfatórios, com pontos fortes e fracos a destacar.

O uso de um único classificador para diferentes funções objetivos de um problema aplicado no algoritmo 3 apresenta bons resultados para a classe de problemas DTLZ, o que ratifica um ganho em empregar a técnica KKTPM como forma de se obter uma nova função discreta com um conjunto finito de soluções. Destaca-se que, para essa classe de problema, as configurações SMPSOSOneAD e SMPSOSOneRF figuram, na maioria das vezes, entre os melhores índices IGDs obtidos para os problemas DTLZ.

Em detrimento aos bons resultados tem-se o baixo desempenho, em termos do índice IGD, para algumas configurações de algoritmos quando aplicadas à classe de problemas WFG. Esse é um ponto em aberto que cabe estudos mais detalhados para encontrar possíveis respostas para o desempenho inferior quando comparado com os resultados para a classe de problemas DTLZ.

Um aspecto importante constatado nos experimentos são os bons resultados obtidos para as configurações que empregam o método de alimentação proposto nos algoritmos 1 e 2. Algumas das configurações que empregam esses algoritmos quando não obtiveram os melhores índices IGDs, conseguiram valores próximos ao índice original do algoritmo original *NSGA-II*.

Outro ponto positivo a se destacar é o emprego de técnicas de aprendizagem de máquina ainda não usadas na literatura na forma de *surrogate*, AD e RF. Além do que, ambas as técnicas mostraram-se com bom potencial em aprender a função objetivo para a classe de problema mono-objetivo e com muito objetivos, independente da abordagem de alimentação usada.

De forma geral, ressalta-se que as abordagens de treino do *surrogate* propostas nessa dissertação, principalmente no contexto de problemas com muito objetivos, possui como pontos positivos a fácil interoperabilidade com diferentes algoritmos evolutivos, assim como diferentes métodos de aprendizagem. Além disso, o uso dessas abordagens proporcionou resultados tão bons quanto os algoritmos original do *NSGA-II* e do estado da arte *MOEADD*, ou até melhores.

Ao final dos experimentos existem alguns pontos que ficam em abertos e serão abordados em trabalhos futuros. São eles, avaliação do tempo de execução das diferentes configurações usando *surrogate*, com isso estima-se identificar se o emprego de *surrogate* em problemas considerados complexos trás ganhos em termos de eficiência. Emprego de abordagens de aprendizagem profunda na forma de *surrogate*, visando um melhor apendizado das funções objetivos, e assim melhorando a eficácia do algoritmo. Avaliar e empregar as diferentes abordagens de alimentação propostas nessa dissetação em um problemas de otimização do mundo real.

Referências

ARGETON, J. L. P. *Árvore de regressão para dados censurados e correlacionados*. Dissertação (Mestrado) — apresentada como requisito à obtenção do grau de mestre, Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica, 2013. Citado na página 24.

ASCIONE, F. et al. Casa, cost-optimal analysis by multi-objective optimisation and artificial neural networks: A new framework for the robust assessment of cost-optimal energy retrofit, feasible for any building. *Energy and Buildings*, Elsevier, v. 146, p. 200–219, 2017. Citado 2 vezes nas páginas 29 e 30.

BEACHKOFSKI, B.; GRANDHI, R. Improved distributed hypercube sampling. In: *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. [S.l.: s.n.], 2002. p. 1274. Citado na página 42.

BRANKE, J. et al. *Multiobjective optimization: Interactive and evolutionary approaches*. [S.l.]: Springer Science & Business Media, 2008. v. 5252. Citado na página 23.

CARVALHO, A. B. *Novas estratégias para otimização por nuvem de partículas aplicadas a problemas com muitos objetivos*. Tese (Doutorado) — apresentada como requisito à obtenção do grau de Doutor, Universidade Federal do Paraná, 2013. Citado 2 vezes nas páginas 21 e 22.

COELLO, C. A. C. Recent results and open problems in evolutionary multiobjective optimization. In: SPRINGER. *International Conference on Theory and Practice of Natural Computing*. [S.l.], 2017. p. 3–21. Citado 3 vezes nas páginas 14, 15 e 28.

COELLO, C. A. C. et al. *Evolutionary algorithms for solving multi-objective problems*. [S.l.]: Springer, 2007. v. 5. Citado 3 vezes nas páginas 20, 21 e 22.

COELLO, C. A. C. et al. *Evolutionary algorithms for solving multi-objective problems*. [S.l.]: Springer, 2007. v. 5. Citado na página 73.

DEB, K.; ABOUHAWWASH, M. An optimality theory-based proximity measure for set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 20, n. 4, p. 515–528, 2016. Citado 2 vezes nas páginas 33 e 47.

DEB, K. et al. Classifying metamodeling methods for evolutionary multi-objective optimization: First results. In: SPRINGER. *International conference on evolutionary multi-criterion optimization*. [S.l.], 2017. p. 160–175. Citado 3 vezes nas páginas 35, 44 e 47.

DEB, K. et al. A taxonomy for metamodeling frameworks for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, 2018. Citado 3 vezes nas páginas 35, 44 e 47.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, v. 18, n. 4, p. 577–601, 2014. Citado na página 22.

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. ISSN 2210-6502. Citado 2 vezes nas páginas 54 e 76.

DURILLO, J. J.; NEBRO, A. J. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, v. 42, p. 760–771, 2011. ISSN 0965-9978. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0965997811001219>. Citado 2 vezes nas páginas 51 e 74.

HAMDAOUI, M. et al. Kriging surrogates for evolutionary multi-objective optimization of cpu intensive sheet metal forming applications. *International Journal of Material Forming*, Springer, v. 8, n. 3, p. 469–480, 2015. Citado 3 vezes nas páginas 30, 31 e 35.

HUSSEIN, R.; DEB, K. A generative kriging surrogate model for constrained and unconstrained multi-objective optimization. In: ACM. *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. [S.l.], 2016. p. 573–580. Citado 3 vezes nas páginas 35, 44 e 47.

ISHIBUCHI, H.; TSUKAMOTO, N.; NOJIMA, Y. Evolutionary many-objective optimization. In: IEEE. *Genetic and Evolving Systems, 2008. GEFS 2008. 3rd International Workshop on*. [S.l.], 2008. p. 47–52. Citado 2 vezes nas páginas 14 e 21.

JIN, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, Elsevier, v. 1, n. 2, p. 61–70, 2011. Citado na página 15.

LIAU, Y. S. et al. Machine learning enhanced multi-objective evolutionary algorithm based on decomposition. In: SPRINGER. *International Conference on Intelligent Data Engineering and Automated Learning*. [S.l.], 2013. p. 553–560. Citado 3 vezes nas páginas 28, 29 e 35.

MATOS, S. A. L. *Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos*. Dissertação (Mestrado) — apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe, 2017. Citado na página 15.

MINING, W. I. D. Data mining: Concepts and techniques. *Morgan Kaufinann*, 2006. Citado 2 vezes nas páginas 25 e 26.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 2 vezes nas páginas 51 e 74.

PORTELLI, G.; PALLEZ, D. Image signal processor parameter tuning with surrogate-assisted particle swarm optimization. In: . [S.l.: s.n.], 2019. Citado 2 vezes nas páginas 33 e 35.

ROSALES-PÉREZ, A. et al. A hybrid surrogate-based approach for evolutionary multi-objective optimization. In: IEEE. *Evolutionary Computation (CEC), 2013 IEEE Congress on*. [S.l.], 2013. p. 2548–2555. Citado 3 vezes nas páginas 28, 29 e 35.

ROSALES-PÉREZ, A. et al. Surrogate-assisted multi-objective model selection for support vector machines. *Neurocomputing*, Elsevier, v. 150, p. 163–172, 2015. Citado 2 vezes nas páginas 28 e 29.

ROY, P.; HUSSEIN, R.; DEB, K. Metamodeling for multimodal selection functions in evolutionary multi-objective optimization. In: ACM. *Proceedings of the Genetic and Evolutionary Computation Conference*. [S.l.], 2017. p. 625–632. Citado 12 vezes nas páginas 7, 16, 22, 29, 30, 31, 32, 35, 42, 43, 44 e 47.

SURJANOVIC, S.; BINGHAM, D. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved october 20, 2018, from <http://www.sfu.ca/~ssurjano>. Citado 2 vezes nas páginas 50 e 52.

VOLZ, V.; RUDOLPH, G.; NAUJOKS, B. Surrogate-assisted partial order-based evolutionary optimisation. In: SPRINGER. *International Conference on Evolutionary Multi-Criterion Optimization*. [S.l.], 2017. p. 639–653. Citado 3 vezes nas páginas 30, 31 e 35.

WANG, W.; PENG, H. A fast multi-objective optimization design method for emergency libration point orbits transfer between the sun–earth and the earth–moon systems. *Aerospace Science and Technology*, Elsevier, v. 63, p. 152–166, 2017. Citado 3 vezes nas páginas 30, 31 e 35.

ZĂVOIANU, A.-C. et al. Multi-objective knowledge-based strategy for process parameter optimization in micro-fluidic chip production. In: IEEE. *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. [S.l.], 2017. p. 1–8. Citado 3 vezes nas páginas 29, 30 e 35.

ZHANG, Y. et al. Multi-objective optimization of double suction centrifugal pump using kriging metamodels. *Advances in Engineering Software*, Elsevier, v. 74, p. 16–26, 2014. Citado 3 vezes nas páginas 30, 31 e 35.