UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE ENGENHARIA DE PRODUÇÃO E GESTÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Wilson Trigueiro de Sousa Júnior

PROPOSTA DE REDUÇÃO DO TEMPO COMPUTACIONAL EM PROBLEMAS DE OTIMIZAÇÃO VIA SIMULAÇÃO A EVENTOS DISCRETOS INTEGRANDO METAHEURÍSTICAS, APRENDIZAGEM DE MÁQUINA E PARALELISMO

Tese submetida ao Programa de Pós-Graduação em Engenharia de Produção do Instituto de Engenharia de Produção e Gestão da Universidade Federal de Itajubá como parte dos requisitos para obtenção do Título de Doutor em Ciências em Engenharia de Produção.

Área de Concentração: Engenharia de Produção

Orientador: Prof. Dr. José Arnaldo Barra Montevechi
Co-orientador: Prof. Dr. Rafael de Carvalho Miranda

Itajubá – MG
2019

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE ENGENHARIA DE PRODUÇÃO E GESTÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Wilson Trigueiro de Sousa Júnior

PROPOSTA DE REDUÇÃO DO TEMPO COMPUTACIONAL EM PROBLEMAS DE OTIMIZAÇÃO VIA SIMULAÇÃO A EVENTOS DISCRETOS INTEGRANDO METAHEURÍSTICAS, APRENDIZAGEM DE MÁQUINA E PARALELISMO

Banca examinadora:

Prof. Dr. David Custódio Sena

Prof. Dr. Robson Bruno Dutra Pereira

Prof.ª Dr.ª Tábata Fernandes Pereira

Prof.ª Dr.ª Mona Liza Moura de Oliveira

Prof. Dr. José Arnaldo Barra Montevechi (Orientador)

Prof. Dr. Rafael de Carvalho Miranda (Co-orientador)

DEDICATÓRIA

A Deus, princípio e fim de todas as coisas, em especial a meus pais, Wilson Trigueiro de Sousa e Maria José Ferro de Sousa, e a minha esposa, Luana Neves Damasceno.

AGRADECIMENTOS

A meus pais, Wilson Trigueiro de Sousa e Maria José Ferro de Sousa, por todo apoio, ensinamentos e exemplos de pessoas durante toda minha vida.

A minha esposa Luana Neves Damasceno, pela paciência, compreensão nos momentos de ausência, incentivar meu desenvolvimento e por sempre estar a meu lado.

Ao professor José Arnaldo Barra Montevechi pela orientação desse trabalho e por toda amizade, auxílio, empenho e conselhos ao longo de meu doutorado.

Aos professores Carlos Mello e Rafael Leme, por todo auxílio prestado como coordenadores do programa de Pós-Graduação em Engenharia de Produção da UNIFEI.

Ao professor e amigo, Rafael de Carvalho Miranda, por toda amizade e contribuições realizadas a esta tese.

Aos professores da Universidade Federal de São João del-Rei, Allexandre Fortes da Silva Reis, Flávio Napolitano, Kívia Mota Nascimento, Leandro Reis Muniz, Lincoln Cardoso Brandão, Roberta Alves e Robson Bruno Dutra Pereira, o meu obrigado pelo apoio em minha instituição de origem.

Aos professores do Núcleo de Estudos Avançados para Auxílio à Decisão (NEAAD), Alexandre Pinho, André Medeiros, Fabiano Leal, José Hamilton e José Antônio de Queiroz.

A todos os meus amigos, em especial Afonso, Fernanda, Flavio, Gustavo, João Paulo, Milena, Mona Liza, Tábata, Tanita e Tiago, por sempre estarem a meu lado no que fosse preciso.

A todos os funcionários da UNIFEI, que direta ou indiretamente ajudaram nesse trabalho.

A CAPES, CNPq e à FAPEMIG pelo apoio e incentivo à pesquisa brasileira, em especial a essa.

Por fim, agradeço a Universidade Federal de Itajubá, a quem sempre terei uma dívida de gratidão.

A todos vocês, meu muito obrigado!

EPÍGRAFE

"*Se cheguei até aqui foi porque me apoiei no ombro dos gigantes*".
Isaac Newton

RESUMO

As técnicas de simulação a eventos discretos têm sido empregadas em diversos setores industriais nas últimas décadas. Isso se tornou possível com a popularização de recursos computacionais e conhecimentos estatísticos aplicados na produção de bens e serviços. Um propósito recorrente ao se gerar uma simulação é a avaliação de uma grande quantidade de cenários, a fim de se encontrar uma relação ótima da combinação de variáveis para se atender restrições e funções de minimização e/ou maximização de objetivos. O processo de otimização dos modelos de simulação tem que lidar com o problema do aumento exponencial do espaço de busca por soluções, enquanto o número de variáveis de decisão cresce linearmente, tornando a resolução desse problema muito difícil ou impossível, no que tange a avaliação de todas as possíveis combinações quando não há tempo e/ou recurso computacional suficientes para tal. Com o aumento da complexidade referente à quantidade de possíveis soluções a serem consideradas pelos agentes tomadores de decisão, a área do conhecimento referente à Pesquisa Operacional tem aplicado técnicas tradicionalmente geradas para resolver problemas de otimização combinatória, em problemas de simulação a eventos discretos. A exemplo de tais técnicas de otimização, as metaheurísticas têm sido utilizadas com sucesso desde a origem dos primeiros métodos de otimização. Um fator restritivo na utilização destes métodos de otimização em simulação a eventos discretos é que para a realização do teste da qualidade de uma dada possível solução, em geral é necessário a utilização de um *software* simulador. Assim, ocorre que mesmo com o uso de método de otimização, muito tempo é dispendido para a obtenção de uma boa solução, pelo fato de ser necessário chamar recursivamente o simulador. Com o intuito de minimizar esse efeito, a presente tese associou quatro abordagens para a diminuição do tempo de resposta necessário para se obter boas soluções para a otimização de modelos de simulação a eventos discretos, utilizando de metamodelagem por aprendizagem de máquina e paralelismo de soluções, associadas a metaheurísticas de busca populacional. Neste contexto, foi possível a integração de todos estes conceitos em um mesmo ambiente, aplicando a mesma em três objetos de estudo referentes a problemas da Engenharia de Produção. Um ambiente de otimização *open source* foi construído em Python para a integração dos três objetos de estudo considerando 33 métodos de aprendizagem de máquina, duas metaheurística e paralelismo do processamento dos cenários. Como resultado médio do método proposto, foi possível a redução do tempo computacional em 93,5% em comparação ao método tradicional, com a utilização apenas de otimização por metaheurística, obtendo uma solução igual a 87,5% do valor de referência dos objetos de estudo.

Palavras Chaves*:* Simulação a eventos discretos; Otimização via simulação; Metaheurística; Aprendizagem de máquina; Paralelismo.

ABSTRACT

Discrete event simulation techniques have been used in several industrial sectors in recent decades, with the advent and popularization of computational resources and statistical knowledge applied in the production of goods and services. A recurring purpose in generating a simulation is the evaluation of a large number of scenarios in order to find an optimal relation of the combination of variables to meet constraints and functions of minimization and/or maximization of objectives. The optimization process of the simulation models has to deal with the problem of the exponential increase of the search space for solutions, while the number of decision variables increases linearly, making the resolution of this problem very difficult or impossible as regards the evaluation of all possible combinations when there is insufficient time and/or computational capability to do so. With the increase in complexity regarding the number of possible solutions to be considered by decision-making agents, the Operational Research area of knowledge has applied techniques traditionally generated to solve problems of combinatorial optimization, in problems of discrete event simulation. For these optimization techniques, metaheuristics have been used with success since the origin time of the first optimization methods. A restrictive factor in the use of these optimization methods in discrete event simulation is that in order to test the quality of a given possible solution, it is generally necessary to use simulator software. Thus, even with the use of optimization method, a lot of time is spent to obtain a good solution because it is necessary to call the simulator recurrently. In order to decrease this effect, the present thesis has associated three approaches to reduce the response time required to obtain good responses in the optimization of discrete event simulation, using machine learning metamodel and solution parallelism within metaheuristics that use population search method. In this context, it was possible to integrate all of these concepts in the same platform, applying the proposed in three objects of study concerning problems of Production Engineering. An open source optimization environment was built in Python to integrate the two study objects with 33 machine learning methods, two metaheuristics, and parallelism for scenario processing. As a mean result of the proposed method it was possible to reduce the computational time by 93.5% compared to the traditional method of using only metaheuristic optimization, obtaining a solution equal to 87.5% of the reference value of the studied objects.

Keywords*: Discrete event simulation; Optimization via Simulation; Metaheuristics; Machine learning; Parallelism.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABC	*Ada Boost Classifier* (Classificador Ada Boost)
AM	*Machine Learning* (Aprendizagem de Máquina)
ANN	*Artificial Neural Networks* (Redes Neurais Artificiais)
ARDR	*Automatic Relevance Determination Regression* (Regressão de Determinação da Relevância Automática)
BC	*Bagging Classifier* (Classificador por Ensacamento)
BNB	*Bernoulli Naive Bayes*
BRR	*Bayesian Ridge Regression* (Regressão por Cume Bayesiano)
CIMO	*Context, Intervation, Mechanism, Outcomes* (Contexto, Intervenção, Mecanismo, Resultados)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
CSV	*Comma Separated Values* (Valores Separados por Vírgula)
DOE	*Design of Experiments* (Delineamento de Experimentos)
DTC	*Decision Trees Classifier* (Classifcador por Árvore de Decisão)
DTR	*Decision Trees Regressor* (Regressor por Árvore de Decisão)
ECOC	*Error-Correcting Output-Codes* (Correção por Erro de Saída)
EM	*Elastic Net* (Rede Elástica)
EP	Engenharia de Produção
ETC	*Extra Trees Classifier* (Classificador por Árvore Extra)
GA	*Genetic Algorithm* (Algoritmo Genético)
GNB	*Gaussian Naive Bayes*
GPR	*Gaussian Process Regressor* (Regressor por Processo Gausiano)
GPU	*Graphic Processing Unit* (Unidade de Processamento Gráfico)
GRASP	*Greedy Randomized Adaptive Search Procedure* (Procedimento de Pesquisa Adaptativa Aleatorizada Gulosa)
GTBR	*Gradient Tree Boosting Regressor* (Regressor por Árvore em Gradiente)
IID	Identicamente e Independentemente Distribuída
KRR	*Kernel Ridge Regression* (Regressão por Cume Kernel)
LASSO	*Least Absolute Shrinkage and Selection Operator* (Menor Encolhimento Absoluto e Seleção de Variáveis)
LDA	*Linear Discriminant Analysis* (Análise Discriminante Linear)
LR	*Logistic Regression* (Regressor Logístico)

MAE	*Mean Absolute Error* (Erro Médio Absoluto)
MLPC	*Multi-layer Perceptron Classifier* (Classificador por Perceptron Multi-Camada)
MLPR	*Multi-layer Perceptron Regressor* (Regressor por Perceptron Multi-Camada)
MNB	*Multinomial Naive Bayes* (Naive Bayes Multinomial)
MSE	*Mean Squared Error* (Erro Médio Quadrático)
NCC	*Nearest Centroid Classifier* (Classificador por Centroide mais próximo)
NNC	*Nearest Neighbors Classification* (Classificador por Vizinho mais Próximo)
NP	Não Polinomial
OE	Objeto de Estudo
OLS	*Ordinary Least Squares* (Mínimos Quadrados Ordinários)
OMP	*Orthogonal Matching Pursuit* (Pesquisa ortogonal de correspondência)
OVOC	*One-Vs-One Classifier* (Classificador Um-Vs-Um)
OVRC	*One-Vs-Rest Classifier* (Classificador Um-Vs-Resto)
OvS	*Optimization via Simulation* (Otimização via Simulação)
PDES	*Parallel Discrete Event Simulation* (Simulação a Eventos Discretos Paralela)
PICO	População, Intervenção, Comparação, Resultados, Contexto
PP	Pergunta da Pesquisa
PR	*Polynomial Regression* (Regressão Polinomial)
RFC	*Random Forest Classifier* (Classificador Floresta Randomica)
RFR	*Random Forest Regressor* (Regressor Floresta Randomica)
RR	*Ridge Regression* (Regressor por Cume)
RRCV	*Ridge Regression with Cross Validation* (Regressor por Cume com Validação Cruzada)
RSL	*Sistematic Literature Review* (Revisão Sistemática da Literatura)
SED	Simulação a Eventos Discretos
SGD	*Stochastic Gradient Descent* (Descida Estocástica de Gradiente)
SPIDER	*Sample, Phenomenon of Interest, Design, Evaluation, Research type* (amostra, o fenômeno de interesse, design, avaliação, tipo de pesquisa)
SVM	*Support Vector Machines* (Máquinas de Vetores de Suporte)
TXT	*Eletronic Text File* (Aquivo Eletrônico de Texto)
VNS	*Variable Neighborhood Search* (Busca em Vizinhança Variável)
WIP	*Work in Process* (Trabalho em Processo)
WSC	*Winter Simulation Conference* (Conferencia de Inverno sobre Simulação)

SUMÁRIO

1. INTRODUÇÃO

Na busca pela manutenção da competitividade das empresas no mercado, a satisfação das exigências constantes dos clientes está na melhoria da qualidade de bens e serviços em relação aos custos e preço final (SALAM; KHAN, 2016). Sendo assim, a gestão de sistemas de produção exige o uso de ferramentas analíticas em muitos casos para ajudar no processo de identificação de oportunidades para melhorar a qualidade geral em termos de produção, logística, e outras questões (DORIGATTI *et al*., 2016; LANDA *et al*., 2018; LOPES *et al*., 2017; SHAHI *et al*., 2016).

Com o advento da indústria 4.0, uma gama de dados está sendo gerada e armazenada em centros de dados com redução de custos relacionados à captura, transmissão e armazenamento de dados. Para que este volume de dados gere um benefício direto para as organizações, são necessárias técnicas que auxiliem em sua análise. Isso deve ser feito de tal forma que os padrões sejam identificados e transformados em informações (ZÚÑIGA; MORIS; SYBERFELDT, 2017).

À luz da questão do processamento de dados, o processamento paralelo é uma técnica que tem sido utilizada, considerando os últimos avanços no que se refere a novas tecnologias de *hardware* para que o tempo computacional não seja um fator proibitivo para uma grande quantidade de análises a serem feitas para com o volume de dados também de grande dimensão, reduzindo assim o tempo necessário para seu processamento (ALBA, 2005; ALBA; LUQUE; NESMACHNOW, 2013; SAVINIEC; SANTOS; COSTA, 2018).

Como exemplo de método para trabalhar com o volume de dados gerados pela indústria 4.0, é possível identificar a Simulação a Eventos Discretos (SED), como um conjunto de técnicas suficientemente robustas no tratamento de dados, captura de padrões e avaliação de cenários para auxiliar no processo de tomada de decisões, integrando métodos estatísticos e gerenciais com essa finalidade (WANG *et al*., 2015; XU *et al*., 2016; ZÚÑIGA; MORIS; SYBERFELDT, 2017; ZÚÑIGA; SYBERFELDT; MORIS, 2017). Da mesma forma, outras técnicas classificadas como “Aprendizado de Máquina”, demanda o conhecimento de mineração de dados e métodos estatísticos, a fim de fazer previsões, classificações e “clusterização” de bases de dados existentes (CALVET *et al*., 2017; JAHANGIRIAN *et al*., 2010).

Para melhorar sistemas de produção considerando dados históricos (de produção), a exemplo pode-se citar problemas relacionados á: alocação de recursos, definição de lote

econômico de produção, determinação de rede de suprimento, etc., como problemas comuns de otimização abordados em projetos de simulação a eventos discretos (LI; JIA; WANG, 2012; LUCIDI *et al*., 2016; SIFALERAS; KONSTANTARAS, 2017; ZSCHIESCHANG *et al*., 2014). Esses estudos buscam a melhor configuração de um dado conjunto de variáveis para aproximar o objetivo de minimizar custos e/ou maximizar o lucro em uma dada situação. A característica de busca dentro de combinações entre os níveis das variáveis (espaço solução) para minimizar os custos e maximizar o lucro é foco do primeiro, segundo e terceiro objetos de estudo avaliados na tese.

Como o conjunto completo de possíveis soluções em tais problemas geralmente exige um tempo exponencial para avaliação, esses problemas são considerados NP (Não Polinomial) - difíceis de resolver. Nesse sentido, algoritmos especiais foram desenvolvidos para obter bons resultados em um tempo razoável (*wall clock*). Esses algoritmos em geral fornecem soluções boas, muito próximas do ótimo local ou global, porém mesmo não sendo o ótimo, são aceitáveis tendo em vista o tempo empregado para resolvê-lo. Para tanto, é possível nomear as classes de heurísticas, metaheurísticas e hiper-heurísticas que representam um grande número do atual campo de pesquisa de otimização (AZIMI; CHARMCHI, 2012; LI; ÖZCAN; JOHN, 2017; MAASHI; KENDALL; ÖZCAN, 2015; RASKA; ULRYCH, 2015; RESENDE; RIBEIRO, 2016).

A presente tese contribui com a área de otimização, propondo e testando um ambiente *open source* desenvolvida em Python 3.6 e aplicada em três objetos de estudo referentes a área de Engenharia de Produção (alocação de recursos e definição do tamanho do lote econômico). Foram reunidos os conceitos de computação paralela, metaheurística e aprendizagem de máquina, beneficiando-se de todos estes conceitos/técnicas de otimização para gerar uma nova abordagem para ajudar na tomada de decisões, envolvendo otimização via SED.

Durante a elaboração da pesquisa, foram identificados na literatura documentos que trabalham com esses conceitos separadamente, mas não juntos, como os trabalhos de Amouzgar, Bandaru e Ng (2018), Bandaru e Ng (2015), e Saviniec, Santos e Costa (2018).

1.1 Objetivos geral e específicos

O objetivo geral da presente tese é propor e testar um ambiente de otimização via simulação a eventos discretos, integrando em um mesmo algoritmo metaheurísticas, aprendizagem de máquina e paralelismo. Para ser aplicado na otimização de modelos de SED que representem problemas da Engenharia de Produção, visando à redução do tempo

computacional necessária para se encontrar boas soluções tão próximas de uma solução ótima de referência.

Como objetivos específicos desta pesquisa, gerados pelo desdobramento do objetivo geral, foi proposto:

- Realizar uma Revisão Sistemática da Literatura (RSL) para justificar a escolha dos métodos de otimização e quais os tipos de objeto de estudo seriam utilizados;
- Propor o método de otimização que englobe as principais técnicas de otimização identificadas na revisão sistemática, de maneira tal que seja possível a interação desses conceitos em um mesmo ambiente computacional;
- Propor e programar um ambiente de otimização, de maneira que seja possível a integração do método de otimização com os objetos de estudo;
- Gerar objetos de estudo a partir da revisão da literatura de maneira que representem problemas da Engenharia de Produção;
- Aplicar o método de otimização proposto dentro do ambiente computacional desenvolvida;
- Avaliar os resultados obtidos com a aplicação do método em objetos de estudos relacionados à problemas de Engenharia de Produção, com a confirmação da redução do tempo computacional.

1.2 Justificativas

A justificativa da escolha e desenvolvimento do tema de Otimização via Simulação (OvS), para a presente tese, se deu pela natureza estocástica das variáveis que vários problemas da Engenharia de Produção têm que lidar para representar a realidade, sendo mais indicado nesse caso, a utilização da simulação a eventos discretos.

Juntando-se à esta natureza o fato de que o conjunto de possíveis soluções dessas variáveis pode formar um espaço solução NP-difícil (assim como definido na seção 1), técnicas de otimização são necessárias para serem integradas à simulação (OvS), com o intuito de encontrar uma boa solução em um espaço de tempo dito ideal para o agente tomador de decisão, que é inferior ao necessário para a avaliação de todas as possíveis soluções.

Três questões de ineditismo reforçam a justificativa para a elaboração da presente tese, contribuindo para a área de OvS. A primeira se relaciona à avaliação da literatura científica e seleção de técnicas que têm mais chances para serem empregadas com sucesso em problemas de otimização via simulação a eventos discretos em problemas de Engenharia de Produção.

Com essa identificação e seleção de métodos, é possível a proposição de novo algoritmo de otimização compostos pela união das técnicas e conceitos relacionados e identificadas na literatura como mais aptos para serem utilizados. O uso da integração de várias abordagens em um mesmo método não foi identificado em nenhuma outra pesquisa de otimização via simulação a eventos discretos até o momento de realização deste trabalho.

A segunda justificativa tem por base a proposição de um ambiente de otimização em que fosse possível a junção destes conceitos mais aptos de otimização em um mesmo ambiente de programação, de fácil acesso a todos os interessados no assunto. Tal ambiente de integração, desses conceitos para a formação de um único ambiente *open source* de otimização via simulação a eventos discretos, não foi encontrada em trabalhos científicos até o presente momento desta pesquisa.

O terceiro ponto de originalidade diz respeito ao teste do novo algoritmo de otimização em três objetos de estudo que, a partir da revisão da literatura, representam problemas da Engenharia de Produção. Os testes nos objetos de estudo selecionados são elaborados de maneira tal que seja avaliado a obtenção de boas respostas com a redução do tempo computacional necessário para a sua determinação, em comparação com a utilização de métodos clássicos de otimização.

1.3 Condições de contorno da pesquisa

O método proposto pela presente tese visa gerar um ambiente computacional de auxílio à tomada de decisão, utilizando da simulação a eventos discretos otimizando problemas de Engenharia de Produção que possuam variáveis discretas, com fator de aleatoriedade e que sejam identicamente e independentemente distribuídas.

O termo “otimização” empregado no trabalho, se refere a problemas que exigem a utilização de técnicas sofisticadas de busca no espaço solução. É comum o emprego do termo em trabalhos que avaliam um reduzido número de soluções, portanto o método desenvolvido na tese se aplica à problema cujo tempo de avaliação de todas as possíveis soluções cresça de maneira polinomial (NP-difícil). Nesse mesmo contexto, foi verificado a utilização da terminologia “otimização por simulação” em que técnicas de SED eram utilizadas também com a avaliação de reduzida quantidade de soluções, e que foge do contexto da presente pesquisa.

Os métodos de otimização utilizados (metaheurísticas, aprendizagem de máquina e paralelismo) possuem uma quantidade de parâmetros que podem ser mudados para que uma melhor aproximação seja gerada aos problemas tratados, possibilitando a obtenção de melhores

respostas ao término do processo de otimização. Com a utilização de mais de um método, para o presente trabalho, torna-se impraticável o teste de todas as possíveis combinações de parâmetros de ajuste dos métodos de otimização, tornando-se este, um problema com espaço solução NP-difícil, no que se refere ao ajuste dos modelos de otimização.

A exemplo, o Algoritmo Genético possui como parâmetros de ajuste o tamanho da população inicial, população final, quantidade de pais selecionados, tipos de método de *crossover*, taxas de mutação, taxas de sobrevivência, critério de parada do algoritmo, etc. Dessa forma a presente pesquisa aborda de maneira mais aprofundada a obtenção dos parâmetros que otimizam os objetos de estudo e não o método de otimização.

O objetivo do trabalho não é o ajuste de um método específico, mas sim o teste do conceito da integração de mais de um método aplicado à otimização de SED e quantificação dos possíveis ganhos computacionais. Desta forma, alguns parâmetros de otimização foram ajustados para serem considerados fixos e outros de maior influência no resultado final foram modificados dentro do escopo do trabalho, deixando como temática para trabalhos futuros a utilização de diferentes níveis de parâmetros dos métodos de otimização tratados na pesquisa.

Os dois primeiros objetos de estudo considerados no trabalho para teste dos algoritmos de otimização, tiveram por base problemas da engenharia de produção, porém foram construídos para atender requisitos específicos desejados. O terceiro objeto de estudo foi retirado da literatura, para comparação dos resultados. Desta forma, foram concebidos para se gerar diferentes tipos de espaço solução com características desejadas, de acordo com o exposto na Seção 6.1.

1.4 Estrutura da tese

A presente tese foi estruturada em seis Capítulos.

O Capítulo 1, já apresentado, discorre sobre os principais temas desta tese, assim como os objetivos a serem alcançados e as condições de contorno que delimitam as suas conclusões.

No Capítulo 2 é utilizada a metodologia de Revisão Sistemática da Literatura para a revisão e relação dos principais conceitos de otimização e simulação a eventos discretos aplicados à Engenharia de Produção, tornando assim a base e justificativa fundamental para a escolha dos métodos de otimização e objetos de estudo e análise do estado da arte no que concerne a essa temática.

No Capítulo 3 é descrito o método de pesquisa adotado, assim como a classificação da pesquisa.

No Capítulo 4 é apresentado o método de otimização via simulação a eventos discretos proposto para a redução do tempo computacional, os recursos computacionais necessários para a sua estruturação e implementação, a descrição dos objetos de estudo que serviram de teste para o método proposto e as métricas de avaliação de desempenho dos algoritmos de otimização.

No Capítulo 5 são apresentados os dados e análise da aplicação do método de otimização nos objetos de estudo, levando em consideração os objetivos do trabalho e as métricas de avaliação definidas no Capítulo 4.

No Capítulo 6 são apresentadas as conclusões da tese com respeito à realização dos objetivos definidos no Capítulo 1, acompanhado pelas considerações finais e proposição de trabalhos futuros.

Em sequência, as Referências Bibliográficas utilizadas para sua elaboração são apresentadas, seguida pelos Apêndices que contêm a relação dos códigos de programação utilizados para o desenvolvimento do trabalho e a listagem dos artigos acadêmicos gerados com a presente tese.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações iniciais

Este Capítulo apresenta o referencial teórico que dá suporte para a presente tese. Pretende-se apresentar o estado da arte em relação aos temas correlatos a este trabalho referentes aos métodos de otimização aplicados em simulação a eventos discretos e apresentar conceitos sobre os métodos de otimização presentes na literatura antes de serem abordados nas Seções 2.6.2 e 2.6.4 em que são apresentados os métodos com maior frequência de utilização e seleção dos que serão abordados no presente estudo.

Para destacar a contribuição científica do presente trabalho, foi realizada uma Revisão Sistemática da Literatura (RSL) nas principais bases de dados nacionais e internacionais no período de 1991 até os artigos publicados em 08/10/2018, de forma a assegurar o ineditismo do tema, a sua relevância para o contexto na área de pesquisa operacional dentro da Engenharia de Produção e uma amostra expressiva dos principais trabalhos já publicados dentro dessa temática.

2.2 Definição de termos para otimização via simulação a eventos discretos

O termo simulação remete ao conjunto de técnicas utilizadas para imitar um comportamento específico de um sistema real ou fictício, usando recursos como tempo e conhecimento, para responder questões elaboradas para a estrutura foco do estudo, quando experimentos são custosos ou impossíveis de serem realizados. Essa metodologia pode ser utilizada em uma variedade de áreas, indústrias e aplicações que, de forma genérica, consiste na coleta de dados e análise com o auxílio de ferramentas computacionais (BANKS et al., 2010; KELTON; SADOWSKI; SWETS, 2010; LAW; KELTON, 1991). As perguntas desenvolvidas são relacionadas com a otimização de características específicas que representam cenários do tipo “e se ...?” para o sistema proposto.

Dessa forma, a otimização é definida como a minimização, maximização ou combinação relacionada a função mono/multi objetivo que resume, de maneira matemática, as perguntas formuladas para o sistema. Assim, diferentes combinações de alternativas são consideradas viáveis, se satisfazem todas as restrições do problema, ou inviáveis se pelo menos uma restrição não é satisfeita. A alternativa que possuir o melhor valor para a função objetivo, sendo viável, é considerada ótima. Se a simulação possuir informação suficiente para representar o sistema analisado, a melhor solução simulada pode ser inferida como ótima e tem

boas chances de ser implementada no sistema real, cumprindo o objetivo de ser uma boa ferramenta de auxílio à tomada de decisão (TAHA, 2007).

Para encontrar a solução ótima, o espaço de busca, que é constituído pela combinação dos possíveis valores das variáveis, é avaliado. O tamanho desse espaço de busca pode ser um problema em termos de recursos necessários para realizar uma busca completa, abrangendo todas as possíveis soluções, para se encontrar a melhor. Os recursos, nesse caso, são comumente relacionados ao poder computacional disponível para desempenhar todas as possíveis soluções, que exige uma quantidade de tempo que o agente tomador de decisão pode não possuir. Estes tipos de problemas são denominados NP-difícil (HE et al., 2017; HERRMANN, 2013; NAWARA; HASSANEIN, 2013). De acordo com Banks (1998), Chen, Jia e Lee (2013) e Xu et al. (2015) um problema de otimização via simulação a eventos discretos (OvS) pode ser formulado como (Equação 1):

$$\max_{x \in X} J(x) = E[L(x; w)] \tag{1}$$

Sendo x é um vetor x-dimensional em que cada posição representa uma variável do problema proveniente do vetor X de restrições definida pelos possíveis valores de x. Como J não pode ser calculado de forma direta, é uma função esperada do vetor x com uma função randômica w que provem da estocasticidade (incerteza) ao sistema em cada rodada completa. Para o restante do trabalho, a sigla "OvS" será usada par designar métodos de otimização aplicados em simulação a eventos discretos, assim como proposto nos trabalhos de Fu (2015) e Miranda, (2015). Um estimador para o valor esperado pode ser obtido pela média apresentada na Equação (2).

$$\bar{J}(x) \equiv \frac{1}{N} \sum_{j=1}^{N} L(x; w_j) \tag{2}$$

Usando da lei dos grandes números e do teorema do limite central, a Equação (2) é um bom estimador para o valor esperado de $L(x; w)$ com a diminuição do desvio padrão amostral quando $N \to \infty$.

Quanto ao tipo de dados coletados, a aleatoriedade dos mesmos possui duas categorias principais: determinístico ou estocástico. A primeira se refere aos dados considerados determinísticos quando não possuem aleatoriedade no tempo, sendo estes estudados pela otimização combinatória (LAROQUE et al., 2012). Quando existe um comportamento aleatório observado durante o tempo de coleta, a simulação é um conjunto de conhecimentos e métodos recomendados para serem utilizados quando o sistema estudado envolva a relação entre as variáveis que os dados coletados possuam: comportamento estocástico, nenhuma ou correlação

mínima e pode ser considerado Independente e Identicamente Distribuído (IID) (BIANCHI et al., 2009). Se uma dessas características não for atendida, os dados devem ser tratados e/ou considerado o uso de outros tipos de técnicas de modelagem e otimização, como otimização linear ou não linear.

Outra característica se refere a como as entidades mudam durante o tempo. Se mudam em pontos específicos do sistema, é considerada discreto (por exemplo, operações como corte, solda e pintura), em oposição as variáveis que mudam continuamente durante um período de tempo como a redução da temperatura de um objeto (ROSSER; SOMMERFELD; TINCHER, 1991). A simulação pode ser utilizada tanto para variáveis contínuas quanto discretas, mas para o presente estudo, será utilizado o conjunto de métodos denominados como simulação a eventos discretos.

Como resultado, a melhor solução é uma consequência de uma grande quantidade de replicações da simulação que, dependendo do tamanho do sistema e do espaço solução, demanda um poder de processamento computacional e de tempo que pode ser proibitivo (FU, 2002).

De uma forma geral, as técnicas de otimização são desenvolvidas para encontrar boas soluções, considerando e passando por soluções viáveis e inviáveis. Dependendo do método, desenvolvem artifícios para escapar de ótimos locais e encontrar soluções próximas ou iguais ao ótimo global ou local conhecido até o momento, de forma que a relação entre tempo e qualidade da solução satisfaça às expectativas dos agentes tomadores de decisão (MAASHI; KENDALL; ÖZCAN, 2015).

2.2.1 Otimização por heurística ou metaheurística

Para resolver os problemas limitados pela definição da Equação (2), muitos autores escreveram sobre o assunto. É possível citar Banks (1998), Chen, Jia e Lee (2013), Dellino e Meloni (2015), Fu (2015), Mujica Mota e Flores De La Mota (2017) e Pawlewski e Greenwood (2014) que dedicaram trabalhos para a implementação de métodos como heurísticas, meta-heurísticas, baseados em gradiente, modelos substitutos e/ou metamodelos, entre outros, aplicados à simulação a eventos discretos.

O estudo de métodos de tomada de decisão feito por Gigerenzer e Gaissmaier (2010) toma a definição de heurísticas como sendo uma estratégia que ignora a parte da informação com o objetivo de tomar decisões de forma mais rápida e com um resultado próximo à resposta da análise de todas as informações. Levando em consideração o conceito anterior, a ciência da

computação utiliza, na área de inteligência computacional, da criação de métodos de busca no espaço solução em que as respostas são construídas e/ou refinadas a partir de uma lógica. Esta lógica está muito atrelada a explorar algum fator específico do problema em questão e que ajuda a determinar uma forma eficiente de se gerar uma boa resposta, que provavelmente não será a melhor para o problema, porém bem próxima dessa solução ótima conhecida (GAVRILAS, 2010).

A exemplo o "problema do caixeiro viajante clássico", em que se pretende construir uma rota de maneira tal a se visitar todas as cidades consideradas, com uma rota que minimiza a distância total percorrida, uma heurística para se construir a solução inicial constitui em ordenar o conjunto de cidades de maneira crescente de acordo com a distância das mesmas para com o ponto de origem. A rota é formada pela inserção da cidade com a menor distância e repete o processo de identificação da cidade mais próxima para a última cidade inserida na rota. Esta heurística é denominada "gulosa" por considerar apenas a melhor solução corrente (GUTIN; YEO; ZVEROVICH, 2002).

Com o término do processo de construção de uma solução inicial, métodos de refinamento podem ser empregados, com o intuito de se encontrar uma solução melhor que a gerada pelo pela fase construtiva. A exemplo do método de "descida/subida" (*descent/uphill*) para refinamento, propõe a seleção de um componente da solução corrente com a geração e avaliação de troca de possíveis soluções, e caso ocorra alguma solução de melhora, a mesma se torna a solução corrente e repete-se o processo de se avaliar vizinhos para outras variáveis selecionadas.

A exemplo do caso do problema do caixeiro, citado no parágrafo anterior, seleciona-se uma cidade da rota e avalia-se a inserção ou troca de cidades próximas à cidade avaliada, e caso exista uma rota com menor distância total, esta passa a ser a solução corrente e passa-se para a avaliação dos vizinhos de outra cidade que fazem parte da solução (VAN BREEDAM, 2001).

Por conceito, as metaheurísticas podem ser consideradas formas genéricas de como o processo de busca no espaço solução deve ser desenvolvido, independente do problema a ser abordado (ao contrário das heurísticas), sem a garantia que o ótimo global seja encontrado, porém em um tempo computacional reduzido e uma boa solução em comparação aos métodos de busca exaustiva (SÖRENSEN, 2015).

As metaheurísticas são agrupadas e definidas de acordo com os seguintes critérios: metaheurística de solução única *versus* baseada em busca populacional; se elas usam memória (das soluções avaliadas ou parâmetros do método e otimização); e se são inspirados na natureza ou fatores biológicos (CALVET et al., 2017).

A exemplo de metaheurística inspirada na natureza, pode-se citar a "colônia de formigas" que é guiada pala busca de uma menor rota entre o formigueiro e a fonte de comida. As formigas são as funções geradoras de solução, em que a resposta para o problema é dividida em níveis. Na medida que as soluções são geradas, o "feromônio" de cada possível solução em cada nível é acrescida quando a solução gerada pela formiga se torna a melhor dentre as outras. Dessa maneira, a melhor rota será formada pelos valores dos níveis que possuírem a maior quantidade feromônio, já que este representa um caminho que esteve presente nas melhores respostas geradas (RAO, 2009).

Por se tratar de uma área de pesquisa que visa a busca dentro de uma grande quantidade de variáveis (NP-difícil), técnicas híbridas de metaheurísticas surgiram para a combinação de mais de um método, a exemplo da:

- *Simheuristic*: Simulação a eventos discretos com otimização por metaheurística (CHICA et al., 2017; JACKSON; TOLUJEVS; REGGELIN, 2018);
- *Learnheuristic*: Combinação de (meta)heurística com métodos de aprendizagem de máquina (CALVET et al., 2017);
- *Hyper-heuristic*: Usa de mais de um método de (meta)heurística em um mesmo método de otimização (BURKE et al., 2013; LI; ÖZCAN; JOHN, 2017)
- Metaheurística paralela: Combinação de metaheurística com distribuição de processamento paralelo e/ou distribuído em um ou mais computadores (ALBA; LUQUE; NESMACHNOW, 2013);
- Simulação paralela: Combinação de processamento paralelo ou distribuído via CPU ou GPU (*Central Processing Unit* - Unidade Central de Processamento e *Graphic Processing Unit* - Unidade de Processamento Gráfico), para a geração de cenário em simulação, podendo ser em um computador, um *cluster* de máquinas ou com a utilização de computadores em local remoto definido como "computação na nuvem" (FUJIMOTO et al., 2017).

A otimização por metaheurística por se tratar de uma forma de busca computacional intensa, outros métodos podem gerar um modelo matemático que represente o comportamento do objeto de estudo inicial, denominados metamodelos (BETTONVIL; DEL CASTILLO; KLEIJNEN, 2009).

2.2.2 Metamodelagem por aprendizagem de máquina

Quando um modelo, em geral matemático ou lógico, consegue expressar com certo grau de confiança relevante (sob a ótica do agente decisor) a relação entre as variáveis de entrada e as de saída (respostas) de um determinado problema, são denominados metamodelos. Estes modelos podem ser preferíveis de serem trabalhados quando: são mais simples de se estudar do que o problema original; compreendem o domínio global das variáveis de interesse; apresentam uma forma explícita; gera respostas determinísticas e são computacionalmente mais eficientes (BARTON, 2009).

Estes metamodelos, também denominados como modelos substitutos, podem ser obtidos de várias maneiras. De acordo com Barton (2009), os passos necessários para se conduzir uma otimização via simulação utilizando-se de metamodelo são:

- Identificar o tipo de metamodelo mais adequado ao problema estudado;
- Planejar um arranjo de experimentos para ajustar o metamodelo;
- Conduzir o arranjo de experimentos planejado no modelo de simulação;
- Ajustar o metamodelo e validar a qualidade de suas previsões;
- Otimizar o metamodelo, ou utilizar o mesmo para gerar uma direção de busca;
- Checar a performance da simulação no ponto de ótimo previsto pelo metamodelo, ou na direção de busca dada pelo metamodelo.

A exemplo, dois dos principais métodos empregados para se gerar metamodelos são as técnicas: estatísticas de regressão (linear e não linear) e Aprendizagem de Máquina por redes neurais artificiais (FU, 2002).

De acordo com Kubat (2017), os métodos de Aprendizagem de Máquina (AM), aqui tratados nesse trabalho, têm a função de gerar algum tipo de previsão com base em dados previamente gerados sobre o problema proposto, com a finalidade de se detectar padrões que estão escondidos em um conjunto de dados original. Os métodos de AM fazem parte de uma área de pesquisa denominada “Ciência de Dados”, tendo o seu termo mais comum em inglês definido como *Data Science*. A Figura 2.1 define as principais áreas do conhecimento necessárias para se desenvolver estudos nessa área.

Figura 2.1 - Gráfico de Venn para a definição das dimensões de um cientista de dados.
Fonte: Adaptado de Ramasubramaniah e Singh (2019)

De acordo com a Figura 2.1 proposta por Ramasubramaniah e Singh (2019), para se gerar trabalhos na área de Ciência de Dados, é necessário três tipos de conhecimento: computacional, matemático e sobre o objeto de estudo. No que se refere ao conhecimento computacional, é necessário conhecer e dominar alguma linguagem/*software* de maneira tal que seja possível a geração, armazenamento, manipulação e tratamento de dados. No que se refere ao campo matemático, métodos de cálculo e geração de índices e métricas estatísticas são necessárias para se avaliar a qualidade das respostas obtidas de maneira tal a se verificar a utilidade dos métodos propostos. E, por fim, o conhecimento e entendimento do problema objeto de estudo é fundamental para se saber se os dados gerados e os resultados obtidos pela aplicação das técnicas são relevantes para o problema proposto.

Ainda analisando a Figura 2.1, as interseções das áreas de conhecimento geram resultados diversos. A interação entre a região de conhecimentos computacionais e o conhecimento do problema gera uma região definida como “Perigo!”. Este fato se deve porque sem a correta interpretação dos dados com base em inferências estatísticas, conclusões errôneas acerca do problema objeto do estudo podem ser geradas. A região entre o conhecimento matemático/estatístico e o conhecimento sobre o problema gera o que foi definido como método tradicional de se gerar estudos a respeito de um problema. Por fim a intercessão entre o conhecimento computacional e o matemático/estatístico, geram os estudos que dão origem aos métodos de AM. Por fim, apenas com a junção de todos esses conhecimentos é que é possível

uma análise completa sobre um determinado objeto de estudo, de acordo com os Cientistas de Dados.

Considerando a Engenharia de Produção, que se utiliza de técnicas estatísticas de controle de processo e a utilização de séries temporais para a inferência de informações, estas mesmas técnicas de Aprendizagem de Máquina estão sendo utilizadas para se extrair informações úteis a partir dos dados gerados em sistemas produtivos. Dessa forma a Engenharia de Produção tem se beneficiados dos avanços gerados pelos Cientistas de Dados, a partir da utilização dessas novas técnicas como ferramentas de auxílio a tomada de decisão no gerenciamento da rotina dos sistemas produtivos.

Para Raschka, Julian e Hearty (2016), os métodos de AM são divididos em três tipos de aprendizagem: supervisionado, não supervisionado e por reforço. Para os métodos de aprendizagem supervisionada, o principal objetivo é o de fazer um modelo aprender a partir de um conjunto inicial de dados previamente rotulado. Nesse caso o termo rotulado significa que a resposta para a interação das variáveis do problema é conhecida. A Figura 2.2 apresenta um fluxograma relacionando os elementos básicos de uma AM supervisionada.

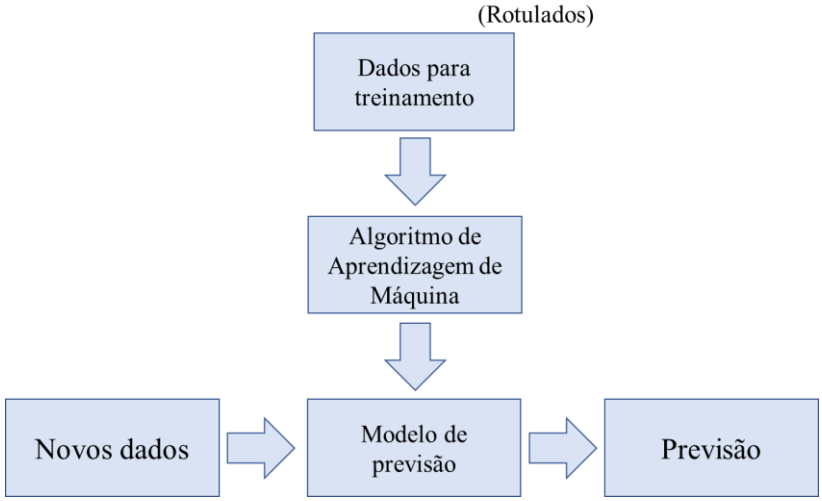

Figura 2.2 - Fluxograma dos elementos de métodos de AM supervisionados.
Fonte: Adaptado de Raschka, Julian e Hearty (2016)

Analisando a Figura 2.2, os métodos supervisionados necessitam de informações das variáveis dependentes e independentes a respeito do objeto de estudo considerado, para se gerar a aprendizagem e/ou ajuste do modelo selecionado. A partir da obtenção de um modelo ajustado, novos dados podem ser testados no modelo ajustado de previsão para se gerar previsões para os novos dados. Considerando os tipos de previsões (variáveis dependentes), os dados podem ser contínuos ou discretos. Caso sejam contínuos, é comum a utilização de métodos de regressão. No caso de previsões discretas, são utilizados métodos definidos como

classificadores, por tentarem definir a classe (valor discreto) que melhor probabilidade tem de representar a resposta para as variáveis independentes consideradas.

Ao se trabalhar com métodos não supervisionados, os dados a *priori* não são rotulados e apresentam uma estrutura desconhecida. A utilização dessa classe de método permite explorar a estrutura dos dados e gerar informação útil sem a orientação de uma variável de resultados conhecida. Um exemplo de método de AM não supervisionado é a clusterização de dados. A Figura 2.3 ilustra um exemplo de clusterização para as variáveis X_1 e X_2 de um problema genérico.

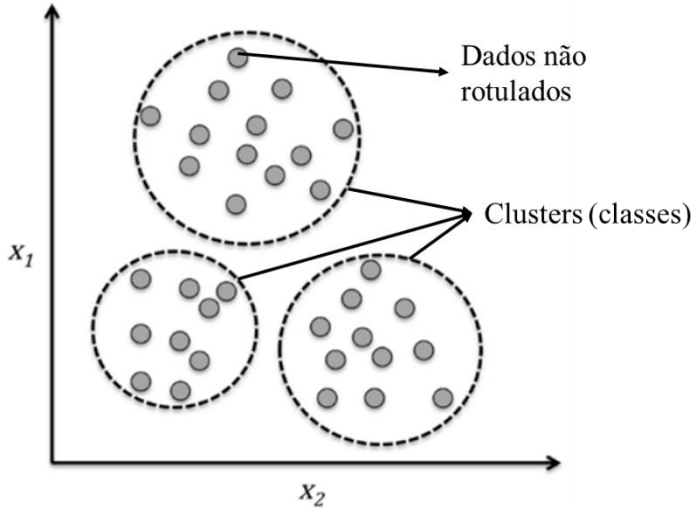

Figura 2.3 - Exemplo de AM não supervisionado por clusterização.
Fonte: Adaptado de Raschka, Julian e Hearty (2016)

Analizando o exemplo da Figura 2.3, o problema consta *a priori* dos dados das variáveis independentes não rotuladas X_1 e X_2. A partir da aplicação do método de clusterização, foi possível a identificação de três classes (*clusters* ou agrupamentos) mais bem definidos, e que de início do problema não eram conhecidos, com base na semelhança das características de X_1 e X_2.

O último tipo de AM está relacionado com os métodos de aprendizagem por reforço. O objetivo nesse tipo de método é a criação de um sistema que constitui a relação do "agente" (método de aprendizagem) com o ambiente. O agente melhora a sua performance tendo por base as suas interações com o ambiente (que neste caso são as alterações das variáveis do problema objeto de estudo). A informação sobre o atual estado do sistema (configuração das variáveis, e resposta para o problema) está associado a um sinal de recompensa que representa quão boa é a resposta para o problema. Através de uma série de interações do agente com o ambiente, por tentativa e erro ou planejamento das ações, através do método de aprendizagem

por reforço escolhido, o agente aprende uma série de ações que maximiza o sinal de recompensa. A Figura 2.4 exemplifica a relação do agente com o ambiente, formando o sistema.

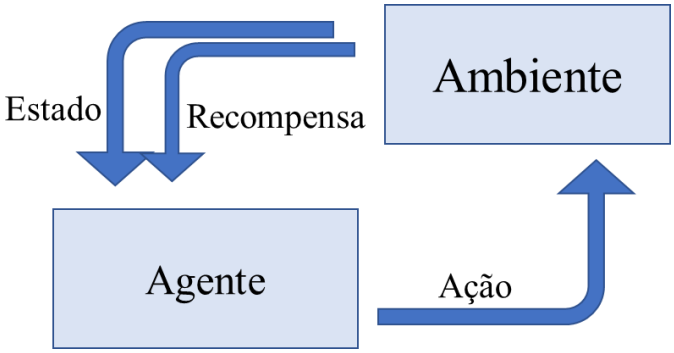

Figura 2.4 - Exemplo de AM por reforço.
Fonte: Adaptado de Raschka, Julian e Hearty (2016)

Analisando a relação ilustrada na Figura 2.4, os métodos de AM por reforço relacionam de forma iterativa com o ambiente até que um critério de parada seja alcançado, e a resposta final gerada está de acordo com a recompensa máxima gerada.

De maneira análoga com os métodos de AM descritos, outro tipo de modelagem e simulação está relacionado ao conceito de modelagem do comportamento da entidade baseado em agentes. Nesse tipo de abordagem de modelagem e simulação, os agentes são indivíduos com comportamento e regras próprios, em que o modelador pode especificar a condição em que as regras serão executadas. Agentes são considerados tomadores de decisão com algum nível de aprendizado e adaptação (COLLIER; NORTH, 2012). Para o presente trabalho foram analisados apenas estudos que envolvem o comportamento de entidades que podem ser explicados por eventos discretos, devido ao fato de que a maioria dos problemas/trabalhos consultados de Engenharia de Produção desenvolvem esse tipo de problema, onde as entidades são transportadas e modificadas de forma específica em processos definidos.

2.2.3 Otimização por paralelismo

Para o processamento tanto de modelos discretos quanto por agentes, é necessário o emprego de recursos computacionais nos quesitos de *software* e *hardware*, e com o surgimento de novas tecnologias no que tange ao *hardware*, estas podem ser incorporadas para serem empregadas em problemas de otimização. Considerando a computação moderna a partir de 1970, alguns fatores são referência para o desempenho de processamento dos computadores, a exemplo da quantidade de transistores, performance por núcleo, frequências dos núcleos e

consumo energético. A Figura 2.5 traz um histórico do desenvolvimento ao longo de 47 anos para as grandezas citadas.

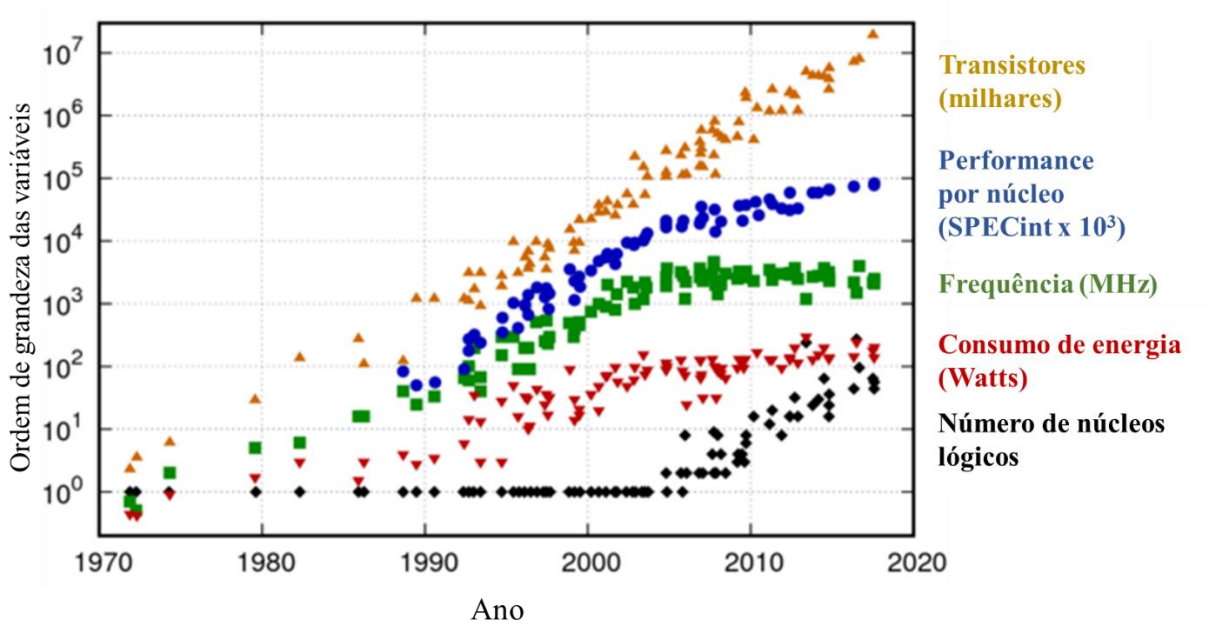

Figura 2.5 - Histórico das variáveis de desempenho da CPU entre 1970 e 2017
Fonte: Adaptado de Rupp (2019)

Analisando a Figura 2.5, é possível observar a evolução de algumas características dos processadores lógicos comerciais ao longo de 42 anos. Com o melhoramento das técnicas de fabricação, foi possível a miniaturização dos transistores, possibilitando o aumento de seu número e por consequência o aumento da performance geral da CPU a uma média do consumo energético. Porém, uma barreira física foi alcançada com a velocidade de processamento chegando próxima de uma barreira física dos 5 GHz. Dessa maneira, uma solução para se aumentar o poder de processamento foi a inserção de maior quantidade de processadores lógicos por processador, gerando as tecnologias *multi-core* (ALBA, 2005).

Com o advento e propagação de processadores *multi-core*, surgiu a possibilidade de se gerar simulações distribuídas e/ou em paralelas, com a utilização de algoritmos divididos basicamente nas classes de conservativos, otimistas ou combinação de ambos, no intuito da execução da simulação em larga escala, utilizando de um ou vários computadores com processadores capazes de processar mais de uma instrução em paralelo via CPU ou com a integração de programação paralela via GPU (COLLIER; NORTH, 2012; FUJIMOTO, 2016).

No presente trabalho, o sentido utilizado de paralelismo foi de se executar vários cenários em paralelo para se diminuir o tempo necessário para se avaliar uma série de diferentes cenários para uma mesma simulação, utilizando da capacidade de processamento paralelo contido em computadores isolados, não na divisão em processamento em *cluster*.

2.3 Revisões da literatura sobre otimização via simulação e temas correlatos

Segundo Maynard e Hodson (2004), a Engenharia de Produção preocupa-se com problemas relacionados à produção de bens e serviços, avaliando os efeitos de projeto, instalação e melhoria de sistemas que integram pessoas, materiais e informações. Salvendy (2001) afirma que esses problemas estão associados à tecnologia, gerenciamento de melhoria de desempenho, gerenciamento, planejamento, projeto e controle de métodos para a tomada de decisão

Para os fins deste trabalho foram considerados todos os problemas envolvidos com a própria produção, no chão-de-fábrica ou nas áreas de produção relacionadas que estão envolvidas com a produção de bens e serviços, sendo considerada uma ferramenta para auxiliar a tomada de decisões na era da indústria 4.0 com meios capazes de lidar com uma grande quantidade de dados para transformar em informações para controle de processos em tempo real (XU et al., 2016).

Durante a pesquisa, 11 artigos sobre revisões da literatura foram encontrados com temas correlacionados a OvS, mas nenhum teve a gama de relacionamento das possibilidades em termos de métodos que foram usados com sucesso, que podem ser considerados no início de projetos de OvS aplicado a Engenharia de Produção. O uso da revisão sistemática da literatura e a lógica CIMO (explicado na Seção 2.5.1) de formulação das questões de pesquisa são uma contribuição científica do presente trabalho.

O primeiro artigo, escrito por Fu (1994a), fez uma revisão sobre os procedimentos de comparação múltipla e procedimentos de classificação e seleção para métodos discretos e baseados em gradiente, método de razão de verossimilhança e experimentação no domínio de frequência para problemas contínuos, aplicados em sistema de inventário (s, S) e os problemas de fila GI / G / 1. Três anos depois, o artigo de Carson e Maria (1997) identificou 6 categorias e 12 métodos de otimização, mas não fez uma relação entre elas e os problemas citados. A próxima revisão da literatura foi conduzida por Ernest et al. (2004), tratando apenas problemas relacionados ao agendamento e escalonamento de equipes. No ano de 2009, Bianchi et al. (2009) realizaram uma pesquisa com base em metaheurísticas de otimização combinatória estocástica.

No artigo de Jahangirian et al. (2010), uma revisão da literatura foi desenvolvida considerando aplicações de simulação em negócios e fabricação entre 1997 e 2006. Em 2013, duas revisões da literatura foram conduzidas nos artigos de Jafer, Liu e Wainer (2013) e Long-

Fei e Le-Yuan (2013), as quais o primeiro considerou o estado da arte em Simulação a Evento Discreto Paralela (em inglês PDES) e o segundo relacionou nove métodos (para variáveis contínuas e discretas), descrevendo o problema de alocação de orçamento. No ano seguinte, a revisão da literatura desenvolvida no artigo de Negahban e Smith (2014) considerou os aspectos do projeto de sistemas de manufatura, operações e desenvolvimento de linguagem / pacote de simulação de 2002 a 2013, com o presente trabalho adicionando a questão do *hardware* para o método da revisão da literatura mencionado anteriormente.

Em 2015, outras três revisões da literatura foram realizadas. A primeira realizada por Xu et al., (2015), baseou-se em seis categorias: classificação e seleção, busca "caixa-preta", metamodelo, métodos baseados em gradientes, caminho amostral, restrições estocásticas e multi-objetivo, explicando cada categoria, com 4 exemplos aleatórios no total. A segunda revisão da literatura, elaborada por Alrabghi e Tiwari (2015), estudou a OvS aplicada ao problema de manutenção. A terceira revisão da literatura (realizada por Bierlaire (2015)) discutiu algumas questões relacionadas ao uso de OvS em problemas de logística. Apenas foi encontrado o artigo de Oliveira, Lima e Montevechi (2016) que declarou de forma expressa a utilização da metodologia de revisão sistemática da literatura relacionando a simulação com a cadeia de suprimentos, mas não especificamente o uso de técnicas de otimização.

Outros seis artigos pesquisados se denominaram como revisões, para aplicação de OvS correlacionadas com a Engenharia de Produção, de uma forma mais específica. Em 1994, Fu (1994b) realizou uma revisão avaliando métodos de otimização em variáveis contínuas e discretas aplicadas a dois exemplos. Kleijnen (2009) revisou a metamodelagem por regressão polinomial de baixa ordem. Em 2013, dois artigos fizeram revisões sobre o assunto. O primeiro, Riley, (2013), citou sete abordagens e alguns casos de aplicação, sem critérios específicos para seleção. O segundo, Alrabghi e Tiwari (2013), referenciou técnicas de otimização baseadas em simulação para operações de manutenção. Em Nguyen, Reiter e Rigo (2014)), foi desenvolvida uma revisão que relatou alguns métodos aplicados a OvS para construir otimização de projetos. Em 2015, Juan et al. (2015) desenvolveram uma revisão que tratou sobre metaheurísticas aplicadas em problemas de simulação estocásticas e combinatória e as diferenças em relação a esses dois campos. A última revisão encontrada na presente pesquisa foi desenvolvida por Kleijnen (2017) e analisou metamodelagem por krigagem em simulação, sem aplicação específica.

Todos as revisões acima mencionadas compuseram a base para criar o método para a presente revisão da literatura aplicado ao tema de OvS, que tem sido estudado nos últimos 27 anos nos artigos selecionados para compor a base de dados onde os dados foram extraídos. Os

tópicos a seguir possuem simularidades com o presente artigo do proponente desta tese: Sousa Junior et al., (2019): Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review.

2.4 A escolha pelo método de Revisão Sistemática da Literatura

Segundo Torraco (2005), a revisão de literatura é uma maneira do pesquisador demonstrar conhecimento em um determinado campo de estudo sobre vocabulário, teorias, variáveis-chave, métodos e história. Como resultado, a revisão de literatura pode ajudar a evitar problemas em todas as etapas, como definição de problema, seleção de método, coleta de dados e análise, o que levará a conclusões de pesquisa com menor probabilidade de os resultados terem falhas ou terem sido mal-entendidos.

Como afirma Denyer e Tranfield (2009), a Revisão Sistemática da Literatura (RSL) não deve ser interpretada como uma revisão de literatura, mas um projeto de pesquisa que utiliza em sua essência a literatura para responder perguntas, de forma que todas as etapas estejam bem definidas e podendo ser reproduzida com viés mínimo, gerando um resultado próximo ao original. Como ponto contrário, alguns autores (HAMMERSLEY, 2001; LEARMONTH; HARDING, 2006; MORRELL, 2008) argumentam que as abordagens baseadas em evidências não são bem recomendadas para ciências sociais e de gestão, outras características como credibilidade local e familiaridade têm importância e afetam diretamente o resultado da decisão tomada. Para corroborar com a ideia da utilização da RSL no presente trabalho, uma pesquisa exploratória por RSL na base de dados Scopus (agosto de 2017) encontrou cerca de 200.000, 7.000 e 5.000 resultados para as áreas de pesquisa de Ciências Médicas, Agrícolas, Biológicas com Engenharia (essas duas ultimas áreas juntas), respectivamente, provando que a metodologia RSL é adequada e utilizada para certas áreas do conhecimento.

De acordo com o *Center for Reviews and Dissemination* (2009), as decisões pertinentes à área da saúde devem ser tomadas com o uso das últimas informações de pesquisa relacionadas às melhores práticas modernas, com a ajuda de uma metodologia que possa unir todas as informações esparsas existentes, por isso a existência de um grande número de RSL apresentados para esta área científica. Em paralelo, as mesmas razões podem ser consideradas para o uso da RSL em Engenharia de Produção (EP), com o benefício que, em geral, as pesquisas da EP não sofrem com os problemas relacionados em Hammersley (2001), Learmonth e Harding (2006) e Morrell (2008) sobre a natureza exata intrínseca deste campo da

ciência, que é beneficiado pelas características da RSL em identificar, avaliar e resumir os dados coletados.

Outras metodologias modernas, como mineração de dados e aprendizado de máquina, podem ser executadas em bancos de dados científicos para procurar trabalhos específicos. A busca automática por palavras exige um conhecimento inicial específico dos termos desejados. Em uma bibliografia diversa e dispersa com temas que progridem no tempo, cada autor utiliza diferentes formas de apresentar sua metodologia e palavras-chave. A busca manual para o presente estudo foi escolhida, em vez de uma busca automática, porque fornece *insights* diretos sobre questões não definidas no início da pesquisa, o que contribui para o desenvolvimento da pesquisa e tendências para trabalhos futuros.

2.5 Aplicação do método de Revisão Sistemática da Literatura

Para realizar a RSL, Booth, Papaioannou e Sutton (2012) referem que a palavra sistêmica implica que a RL deve ser realizada com as seguintes características: explícita, transparente, metódica, objetiva, padronizada, estruturada e reprodutível. A partir deste objetivo, a definição dos passos deve ser feita com cuidado. Neste trabalho foram considerados os passos apresentados em Oliveira, Lima e Montevechi (2016), que consiste em (1) planejamento, busca / (2) triagem, análise / (3) síntese e apresentação dos resultados.

2.5.1 Planejamento da Revisão Sistemática da Literatura

A primeira fase foi responsável por realizar uma melhor compreensão das questões centrais relacionadas à pesquisa em si. Uma busca exploratória foi realizada nas bases de dados *Web of Science* e *Scopus* por se tratarem de bases que possuem uma taxa média de omissão de citações entre 5 e 10% do valor total real das citações (FRANCESCHINI; MAISANO; MASTROGIACOMO, 2014). Depois disso, foram feitas discussões e reuniões sobre os métodos e assuntos relacionados a OvS. Para este propósito, duas reuniões foram realizadas com a participação de três professores com experiência na prática e teoria de OvS, e duas reuniões foram feitas com a colaboração de um professor, um doutor e mestres estudantes que pesquisam neste mesmo campo.

As reuniões definiram a busca por artigos e conferências utilizando os termos *discrete event simulation*, lógica booleana "AND/OR" com os termos: *otimization*, DEA, *metaheuristic, evolutionary, tabu, design of experiments, response surface, metamodel* e *parallel*. Como

resultado da pesquisa exploratória nas bases de dados *Web of Science* e *Scopus*, a Figura 2.6 resume os níveis a serem considerados na OvS.

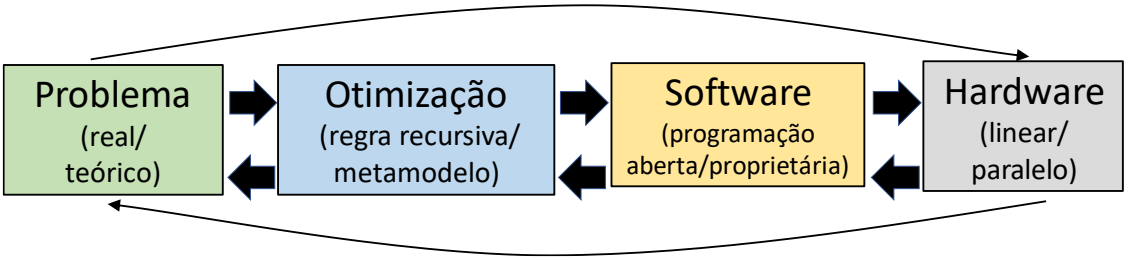

Figura 2.6 - Diferentes níveis iniciais a considerar em projetos de OvS.
Fonte: O autor

A Figura 2.6 representa os primeiros achados para a triagem exploratória nas bases de dados selecionadas. O *design* para a referida Figura 2.6, foi pensado para ser ao mesmo tempo iterativo e recursivo, onde o pesquisador ou praticante possa começar o seu projeto de OvS a partir do problema desejado, escolher o melhor método de OvS que é conhecido por ele/ela. Então, o software e o hardware são escolhidos de acordo com os recursos disponíveis. A análise pode ser tanto iterativa e recursiva porque, na fase inicial de projetos OvS, a seleção do método de otimização é condicionada aos recursos disponíveis em termos de licença de *software*, poder computacional e conhecimento que precisam ser desenvolvidos, exigindo tempo e dinheiro para a aquisição e desenvolvimento. A proposta em desenvolver a RSL consiste em:

a) Desenvolver uma ampla RSL sobre a aplicação da OvS em EP;

b) Identificar e extrair os métodos encontrados nos trabalhos aplicados para resolver OvS em problemas de EP;

c) Analisar e resumir os métodos encontrados;

d) Discutir as suposições de acordo com os resultados encontrados.

Para atingir essa proposta, Perguntas da Pesquisa (PP) foram formuladas para ajudar na extração de dados válidos dos artigos. O Centre for Reviews and Dissemination (2009) define que as PPs devem ser definidas em termo de: população, intervenções, comparadores, resultados e, se a pesquisa exigir, desenho do estudo. Este método é conhecido pelo acrônimo PICO ou PICOS (em português População, Intervenção, Comparação, Resultados, Contexto), geralmente usado como RSL em estudos médicos.

Outros *frameworks* para a medicina foram encontrados, como no trabalho de Booth, Papaioannou e Sutton (2012), com o uso do método SPIDER (*sample, the phenomenon of interest, design, evaluation, research type*, que em português significa amostra, o fenômeno de interesse, *design*, avaliação, tipo de pesquisa) (METHLEY et al., 2014). De acordo com Denyer,

Tranfield e Aken, (2008), os dados para estudos de organização e gestão (podendo ser incluindo a EP nessa análise) são fragmentados e precisam de um *framework* específico.

Nesse contexto, propõe-se que a utilização da lógica CIMO envolva o problema em um Contexto (do inglês *Context*) que necessita de uma Intervenção (do inglês *Intervation*) específica e use um Mecanismo (do inglês *Mechanism*) para gerar Resultados (do inglês *Outcomes*). Outros estudos usaram a lógica CIMO para expressar as PPs (COSTA; SOARES; DE SOUSA, 2016; KRAUSE; SCHUTTE, 2016; PILBEAM; ALVAREZ; WILSON, 2012; RAJWANI; LIEDONG, 2015; TANSKANEN et al., 2017). Para o propósito do presente artigo, as PPs utilizadas foram definidas de acordo com a lógica CIMO que foi dividida em quatro partes:

- Contexto: quais problemas reais ou teóricos da Engenharia de Produção…
- Intervenção:… usa um algoritmo de otimização…
- Mecanismo:… combinado com simulação a eventos discretos…
- Resultados:… para encontrar a melhor solução em termos de qualidade definida nos objetivos do problema e recursos do projeto.

As PPs foram definidas para descrever as características de cada uma das dimensões exploradas pela lógica CIMO, sendo o item "Mecanismo" definido como a simulação a eventos discretos um pré-requisito básico para as PPs. As questões foram definidas como:

- PP1: Quais os principais problemas estudados, relacionados à área de Engenharia de Produção (Contexto)?
- PP2: Quais métodos de otimização e software de implementação foram os mais utilizados (Intervenção)?
- PP3: Como os resultados foram medidos (Resultados)?
- PP4: Qual autor, universidade, ano de publicação e revista foram encontrados que compõem os centros de pesquisa de referência na área de OvS (Contexto)?

Para responder às PPs, a Figura 2.7 resume os dados que foram coletados nos artigos para se acoplar aos objetivos, formando os pilares da pesquisa.

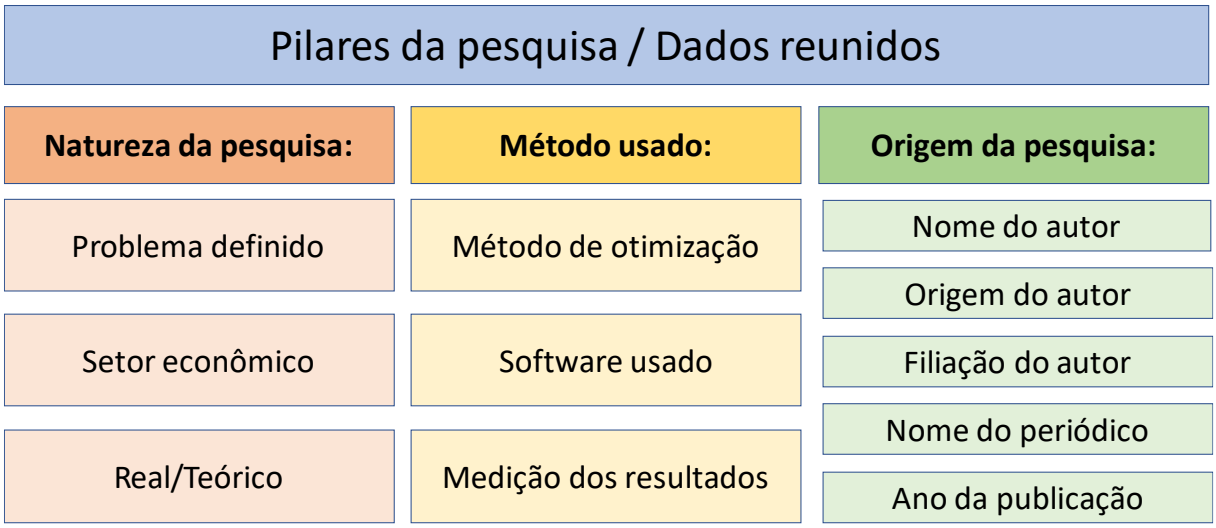

Figura 2.7 - Pilares da pesquisa e dados coletados nos artigos
Fonte: O autor

A Figura 2.7 ilustra as três áreas que foram consideradas para a extração de dados dos artigos. A primeira área é a natureza da pesquisa que define a categoria do problema, o setor da economia e se o projeto é baseado na solução de um problema real ou teórico com dados extraídos da literatura. A segunda categoria representa os métodos usados para executar a OvS, qual o método de otimização, o *software* usado e como os resultados foram medidos. A última categoria considera as informações disponíveis da equipe de pesquisa nos artigos de acordo com o nome do autor, onde ele trabalha (universidade ou empresa), o país de origem e os dados do periódico referente ao nome e ano de publicação.

2.5.2 Pesquisa e triagem da RSL

Para a realização de uma pesquisa ampla e evitar o efeito de omitir possíveis citações relevantes (FRANCESCHINI; MAISANO; MASTROGIACOMO, 2014), as seguintes 18 bases de dados foram selecionadas para a utilização de seus próprios mecanismos de busca acadêmicos: *ACM Digital Library*, *CiteSeerX*, *dblp* Bibliografia da Ciência da Computação, Diretório de Revistas de Acesso Aberto (DOAJ), *Esmerald Insight*, *Google Scholar*, *IEEE Xplore*, Pesquisa acadêmica da *Microsoft*, Portal Capes, *Research Gate*, *Sage Journals*, *Scielo*, *Science Direct*, *Scopus*, *Semantic Scholar*, *Springer Link*, *Web of Science* e *Wiley Online Library*. As bases de dados foram consultadas na mesma ordem alfabética apresentada.

Após a discussão para a busca exploratória, as palavras-chave *Discrete Event Simulation* foram selecionadas com lógica booleana (AND/OR) com: *Optimization*, *Metaheuristic*; *Genetic*; VNS; GRASP; Tabu; *Particle swarm*; *Ant colony*; *Design of experiments*; *Response*

surface; *Factorial*; *Metamodel*; *Model reduction*; *Parallel*; *and* GPU, realizando 15 pesquisas para cada um dos 18 mecanismos de busca acadêmicos apresentados, gerando um total de 270 buscas. Os resultados dos mecanismos de pesquisa são classificados pelos critérios de relevância, o que significa que a lista de resultados presente nas primeiras posições de artigos possui todos os termos seguidos por menos de uma das palavras-chave de pesquisa e assim por diante.

A leitura das listas de resultados das pesquisas nas bases consiste em verificar o título, resumo e, quando possível, ler diretamente os resultados. Se o artigo ou conferência for: relevante, usar o idioma inglês, for revisado por pares e ter sido publicado a partir de 1991 e ser possível fazer o *download* do artigo completo, o mesmo é baixado e catalogado utilizando o programa de referência bibliográfica *Mendeley* versão 1.19.2. O critério de parada para pesquisar nos resultados específicos da combinação de palavras-chave selecionada (mais de 200.000 na estimativa), para cada base de dados, foi quando a posição do último artigo de *download* tinha pelo menos 20 posições à frente do atual artigo analisado. A Tabela 2.1 resume o número de artigos baixados, nas bases de dados consideradas.

Tabela 2.1 - Artigos baixados das bases de dados

Base de dados / mecanismo de busca	**Baixados**
ACM Digital Library	223
Web of Science	172
IEEE Xplore	170
Sage Journals	97
dblp Computer Science Bibliography	52
Semantic Scholar	40
Emerald Insight	38
Directory of Open Access Journals	28
Science Direct	24
Scopus	26
Scielo	10
Springer Link	4
Total	**885**

A Tabela 2.1 apresenta a distribuição do total de 885 (363 + 522, explicado na Figura 2.8) artigos baixados de 12 bancos de dados/mecanismos de busca, mostrando o resultado e classificando as bases de dados mais prováveis para encontrar artigos de *journals* e artigos de conferência relacionados a OvS. O resultado apresentado é parcialmente influenciado pelo fato de que a consulta nas bases de dados foi feita na mesma ordem apresentada na Tabela 2.1 e os nomes repetidos foram descartados, o que significa que os últimos nomes não podem ser considerados piores, por exemplo, *Scopus* melhor do que ACM porque tinha 17 diferentes do

primeiro. Os bancos de dados *CiteSeerX*, Google Acadêmico, Pesquisa Acadêmica *Microsoft*, Portal Capes, *Research Gate* e *Wiley Online Library* não retornam resultados significativos para as combinações propostas de palavras-chave de pesquisa. A Figura 2.8 lista o processo para os artigos selecionados.

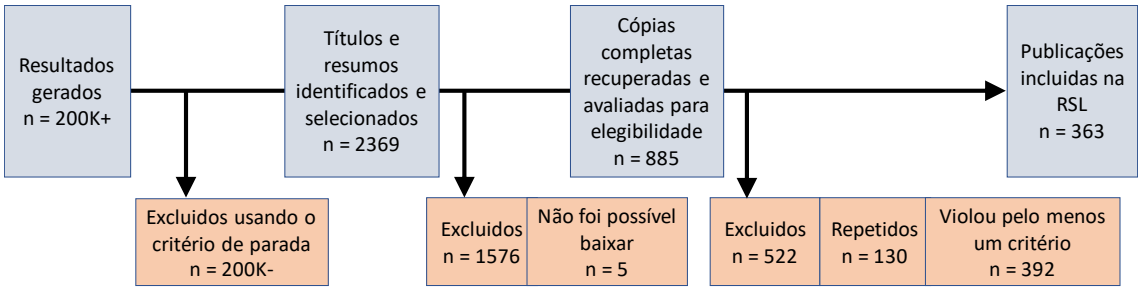

Figura 2.8 - Pesquisa e triagem da RSL
Fonte: O autor

A Figura 2.8 resume o processo de pesquisa e triagem com a combinação proposta de palavras-chave e bancos de dados selecionados. Neste contexto, não foi considerado o procedimento de inclusão para ler e pesquisar na seção de referência bibliográfica de todas as publicações selecionadas (apenas em artigos relevantes para a pesquisa), motivado pela ampla pesquisa e pelo já grande número de artigos repetidos gerados por essa metodologia. Após o uso dos critérios de parada e triagem, 885 artigos foram baixados e 522 foram excluídos por uma ou mais combinações da seguinte regra: 62 (15,86%) eram teóricos, 202 (51,66%) não tinham problema de EP, 77 (19,69%) não possuem um arcabouço de SED ou utilizam entidade baseada em agente, 86 (21,99%) não utilizam um método de otimização válido, 42 (10,74%) usam variáveis determinísticas e 2 (0,51%) foram escritos em um idioma não inglês.

Para identificar os 130 arquivos repetidos, os 885 artigos baixados foram reunidos na mesma pasta em que cada arquivo foi rotulado com o nome do artigo e organizado em ordem alfabética. Esse fato ajudou porque cada banco de dados de pesquisa tem seu próprio caminho para rotular os arquivos, dificultando a identificação dos repetidos. Quando dois arquivos consecutivos tinham o mesmo nome e tamanho de arquivo, eles eram avaliados se pudessem ser repetidos. Posteriormente, o software *Mendeley* possui a função de verificação de artigos duplicados, função também utilizada e que gerou resultado igual ao identificado anteriormente.

Por estudos teóricos excluídos deste trabalho, foram considerados artigos que discutem apenas uma parte específica do processo de simulação e não possuem um estudo de caso ou pesquisa ação (e.x. Adegoke, Togo e Traore, 2013; Thomas, Howes e Luk, 2009). Artigos que não tiveram um problema de EP foram considerados com tópicos como desenvolvimento de

software, sistemas biológicos e tráfego urbano (e.x. Montagna, Viroli e Roli, 2015; Muta et al., 2015).

A exclusão de artigos não se deu pela área, mas sim pela dependência da aplicação para tais áreas, por exemplo a área de manutenção pode realizar estudos preventivos, preditivos e corretivos nos quais a EP terá pouco conhecimento para trabalhar, mas relacionado ao problema de *timetabling* do pessoal de manutenção, um EP pode executar uma OvS com a assistência e dados do caso real.

Neste estudo, a SED é considerada uma metodologia com seu próprio corpo de conhecimentos que requer etapas como: formulação de problemas, coleta de dados, modelagem, verificação, validação, experimentos e conclusão. Para realizar uma OvS, artigos como Iles, Deugo e Canada (2002) consideram a simulação um problema combinatório como proposto em Kleijnen (2009). Neste mesmo sentido, o uso do termo *simulação* para diferentes tipos de projetos, a simulação baseada em agentes não foi considerada na presente pesquisa, demandando um trabalho futuro específico sobre o tema.

2.5.3 Análise, síntese e apresentação dos resultados da RSL

Considerando a síntese dos dados coletados, utilizou-se a planilha eletrônica *Microsoft Excel*® para compilar todas as informações extraídas dos 363 artigos selecionados. Para responder aos PPs, cada artigo selecionado foi avaliado nos seguintes 10 itens: qual o problema; tipo de estudo de caso; método de otimização, software da implementação; medição de resultados; nome do autor; ano da publicação; nome da publicação; afiliação do autor e nacionalidade, retomando a proposição ilustrada na Figura 2.6. Após a coleta de dados, termos semelhantes foram identificados e reunidos para uma melhor síntese, por exemplo, metamodelo e meta-modelo ou delineamento de experimentos.

Após a síntese dos dados nos grupos apresentados na Figura 2.6, o Excel® foi utilizado para estatística descritiva para determinar a porcentagem para cada tipo de problema, método e origem da pesquisa. Esta síntese de dados foi a base para a análise da RSL que consistiu em apresentar os resultados e as melhores práticas para o desenvolvimento da OvS em projetos de EP.

Os 363 artigos foram divididos em duas categorias: periódicos e anais de congressos internacionais, respectivamente 209 e 154. A partir dessa separação, os resultados foram divididos para auxiliar na identificação da fonte de pesquisa.

Para apresentar os resultados após o processo de análise e síntese, as Tabelas 2.2 a 2.5 e as Figuras 2.9 e 2.10 resumem os dados. Esses resultados foram apresentados em sequência para responder as PPs, desenvolvendo uma discussão pelos autores para apresentar as técnicas mais atuais que estão sendo aplicadas aos projetos de OvS sobre os problemas de EP, mostrando as práticas passadas e presentes, trazendo possibilidades para o futuro da OvS em EP, fazendo parte assim do embasamento teórico do presente trabalho.

2.6 Compilação dos dados e discussão

A fim de apresentar os resultados para responder às questões de pesquisa, os dados reunidos para os pilares da pesquisa são apresentados nesta seção para gerar as bases para a discussão.

2.6.1 Natureza da pesquisa

Esta seção apresenta os resultados relacionados ao primeiro pilar da pesquisa "natureza da pesquisa" que está relacionada à informação do problema abordado, ao setor econômico e à definição se é real ou teórico. Para responder a PP1, a Tabela 2.2 resume as descobertas dos problemas encontrados.

Tabela 2.2 - Tipo e setor econômico do problema

Principais áreas e total de artigos encontrados					
Tipo de problema	**Artigos**	**Congressos**	**Total**	**%**	**Cumulativo %**
Programação da produção	74	61	135	36,8	36,8
Chão de fábrica	66	44	110	30,0	66,8
Logística	32	31	63	17,2	83,9
Controle de Estoque	20	19	39	10,6	94,6
Não especificado	4	16	20	5,4	100,0
			367		
Setor econômico					
Secundário	118	90	208	56,7	56,7
Terciário	61	47	108	29,4	86,1
Não especificado	23	25	48	13,1	99,2
Primário	2	1	3	0,8	100,0
			367		
Setor produtivo					
Não especificado	132	125	257	70,0	70,0
Outros	18	14	32	8,7	78,7
Semicondutor	12	14	26	7,1	85,8
Saúde	15	11	26	7,1	92,9
Automotivo	12	9	21	5,7	98,6
Químico	4	1	5	1,4	100,0
			367		
Origem dos dados					
Real	116	98	116	58,3	58,3
Teórico	82	71	82	41,7	100,0
			367		

A Tabela 2.2 resume os problemas encontrados em OvS aplicado em EP, separados em três categorias principais: o tipo de problema, o setor econômico de produção e a indústria relacionada. No tipo de problema, foram geradas cinco categorias de acordo com o número expressivo de artigos relacionados (no mínimo 20) denominados: chão de fábrica (*shop flor*), controle de estoque, programação da produção (*scheduling*), logística, e não especificado para separar os artigos. Para o processo industrial, estão relacionados problemas que ocorrem ou estão relacionados principalmente ao chão de fábrica de uma indústria, onde parte do valor é gerado.

Tais problemas estão relacionados à definição de parâmetros da produção (AL-AOMAR; AL-OKAILY, 2006; CAN; HEAVEY, 2011; CHOI; SEO; KIM, 2014; POWELL, 2018), *buffer* como área de absorção de variabilidade entre processos (LANDA et al., 2018), inspeção (ALRABGHI; TIWARI, 2015; VAN VOLSEM; DULLAERT; VAN LANDEGHEM, 2007), e alocação de recursos (LUCIDI et al., 2016).

A soma dos artigos é maior do que a quantidade original (369 > 363), porque os artigos apresentaram mais de um problema relacionado a EP (CAN; HEAVEY, 2012). Os outros tipos de problemas estão relacionados aos sistemas de apoio à produção, principalmente no que diz respeito ao controle de estoque (KILMER; SMITH; SHUMAN, 1999) para o nível de estoque e reabastecimento; programação associada à programação de *job shop* e despacho (K.T.; PRABAGARAN; JOSEPH, 2017; NASIRI; YAZDANPARAST; JOLAI, 2017; NGUYEN; ZHANG; TAN, 2018; PHANDEN; SAHARAN; ERKOYUNCU, 2018); logística para alocação, localização, *layout*, fornecimento e problemas de roteamento ou estudos não especificados (BARLOW et al., 2018; NOORDHOEK et al., 2018; POETING et al., 2017) que não pode ser determinado em uma das categorias de visualizações.

Os quatro principais problemas, identificados como: programação da produção, processo industrial, logística e controle de estoque, representam 94,6% do total. Esse é um sinal de que esses objetos de estudo possuem variáveis aleatórias e IID, constituindo um espaço de busca NP-difícil que demanda uma metodologia sofisticada, como OvS, para auxiliar no processo de decisão para encontrar a melhor solução, como consequência direta que esses problemas têm nas operações das empresas, assim como definido na Seção 2.2.

A presente RSL não procura buscar outros tipos de metodologias para avaliar esse tipo de problema, mas é possível inferir que, para tais estudos, a OvS é uma ferramenta viável a ser considerada e usada. O tamanho dos espaços soluções apresentados foi em geral composto por poucas estações e/ou *buffers* (1 a 5), representando apenas uma parte dos sistemas de produção

total, mostrando que a OvS foi planejada para resolver parte de um objetivo específico, e não para avaliar todo o sistema. Para os problemas exibidos foi comum a apresentação do tamanho dos espaços de busca que justifica o uso de métodos de otimização.

Para o setor econômico, as quatro categorias foram: primária, relacionada à produção ou exploração de recursos naturais; secundária, responsável pela transformação dos recursos naturais em bens; terciária, associada à prestação de serviços, e não especificado quando o objeto de estudo não pode ser detalhado em uma das outras categorias. Como esperado, o mesmo sistema de produção pode ter uma perspectiva de fornecer bens e serviços ao cliente. Neste caso, foi considerado o objetivo principal do problema. Neste contexto, por exemplo, uma programação de *job shop* e problemas na cadeia de suprimentos relacionados à indústria automotiva estão, respectivamente, relacionados ao setor econômico secundário, tendo como objetivo principal a produção e provisão de um bem.

Foram encontrados apenas quatro trabalhos para o setor primário, representando 0,8% do total (Nageshwaraniyer, 2018; Nageshwaraniyer, Son e Dessureault, 2013a; Nageshwaraniyer, Son e Dessureault, 2013b; Upadhyay e Askari-Nasab, 2018), referente a uma mina de carvão e outra de cobre. Isso pode ser devido ao fato de que os problemas de OvS são estudados principalmente em cursos de engenharia específicos, como Industrial, Produção, Mecânica, Elétrica e Computação. Outros cursos, como Agronomia, Zootecnia e Engenharia de Minas, têm um enfoque diferente e podem depender do conhecimento dos cinco cursos apresentados anteriormente. Este fato sugere áreas a serem exploradas para trabalhos futuros, com a colaboração entre os profissionais formados nos cursos apresentados.

O setor secundário estava presente na maioria dos casos, em 56,7% dos artigos. Vale aqui notar que as informações para determinar se o artigo apresentado estava relacionado ao setor secundário, em alguns casos foram feitas por meio de inferência indireta, quando foram apresentadas informações referentes à manufatura ou produção, ou sugerindo que o objeto de estudo estava relacionado, em 29,4% dos artigos, ao setor terciário. Esse desequilíbrio entre o secundário e o terciário pode ser uma evidência de que a produção de bens é mais adequada para ser modelada pelos métodos de SED, devido ao fato de que a produção de um serviço é mais dependente da interação entre cliente e provedor, que tem fatores humanos e características culturais que são difíceis de serem modeladas por SED e pode ser um campo a ser explorado pela metodologia de simulação baseada em agentes (DORIGATTI et al., 2016).

O terceiro dado reunido na Tabela 2.2 foi o tipo de indústria em que o problema estava relacionado com as indústrias de semicondutores, saúde, automotiva e química, com a soma representando 21,3% do total. Essas indústrias estão relacionadas aos setores econômicos

secundário e terciário e são consideradas como produtoras de bens e serviços de alto nível de valor agregado, em comparação com as indústrias do setor primário.

A categoria "outros" se refere a uma variedade esparsa de indústrias que representam 8,7% do total. A categoria "não especificada" está relacionada aos artigos em que não foram possíveis a identificação da indústria, representando 70,0% do total. O fato de o artigo não especificar o setor econômico a que se refere pode ser por uma necessidade de ocultar do público o nome da empresa que forneceu os dados. Nesse sentido, para auxiliar os trabalhos futuros, a especificação do objeto que gerou o problema em um primeiro momento pode ajudar os profissionais e pesquisadores a focalizar a busca para encontrar soluções já implementadas para um caso específico, e evitar os processos de tentativa e erro de um método genérico. Essa questão da origem dos dados foi observada tanto em artigos de periódicos quanto anais de congresso.

A última informação apresentada na Tabela 2.2 foi a origem dos dados utilizados nos objetos de estudo dos artigos. Por artigos teóricos, foram considerados os que utilizaram dados de outros estudos que, principalmente, não foram gerados pelos autores, ou utilizaram problemas clássicos apresentados em livros ou literatura especializada. Nesta questão, é mostrado um equilíbrio relativo entre o número de publicações em ambas as direções, tanto em periódicos como em congressos. Isso pode ser interpretado como o desenvolvimento da OvS sendo realizada de maneira teórica e prática de forma conjunta.

Respondendo a PP1 "Quais os principais problemas estudados, relacionados à área de Engenharia de Produção?" É possível, segundo os dados apresentados na Tabela 2.2 e a sua correspondente interpretação apresentada subsequentemente, inferir que os setores industriais responsáveis pela geração de bens e serviços de produção, com alto valor agregado, respondem pelos projetos que mais investiram na SED e buscaram a solução otimizada dos problemas apresentados, as quais em sua maioria era relacionada aos problemas de custos e lucros industriais, tais como programação da produção, processo industrial, logística e controle de estoque.

2.6.2 Métodos usados nos artigos pesquisados

Para "caminhar" no espaço solução, procurando a melhor solução viável, uma variedade de métodos de otimização pode ser aplicada a um problema de OvS. A Tabela 2.3 resume os métodos encontrados para otimizar os problemas apresentados na Seção 2.6.1.

Tabela 2.3 - Métodos de OvS identificados

			Total e % do total		
Método de otimização:	**Artigos**	**Congressos**	**Total**	**%**	**Cumulativo %**
Heurística					
Busca local	7	3	10	16,7	16,7
Busca randômica	5	2	7	11,7	28,3
Escalada	3	2	5	8,3	36,7
Outros	24	14	38	63,3	100,0
			60		
Metaheurística					
Genético	77	69	146	42,6	42,6
Arrefecimento simulado	17	16	33	9,6	52,2
Busca tabu	16	8	24	7,0	59,2
VNS	5	1	6	1,7	60,9
Outros	96	38	134	39,1	100,0
			343		
Modelos substitutos					
Fatorial	22	17	39	25,5	25,5
Superfície de resposta	14	10	24	15,7	41,2
ANN	14	3	17	11,1	52,3
Regressão	10	6	16	10,5	62,7
Krigagem	8	5	13	8,5	71,2
DEA	7	0	7	4,6	75,8
Outro	15	22	37	24,2	100
			153		
Paralelo / distribuído					
CPU	9	19	28	77,8	77,8
GPU/CPU	4	4	8	22,2	100
			36		
Proprietário	16	13	27		100
Monte Carlo	6	3	9		100
Baseado em Gradiente	11	5	16		100
Outros	38	36	74		100

Foi observada uma prática comum nos artigos, como o uso de mais de um modelo de otimização no mesmo trabalho, justificando assim o total de 718 métodos e implementações encontrados na Tabela 2.3. Outro fator é que um método/artigo pode usar uma mistura entre dois tipos diferentes de modelagem, como programação inteira, binária ou mista (SAREMI et al., 2013). Nesses casos foram considerados dois métodos, de acordo com a maioria das implementações, e pelo menos a presença em cinco artigos para serem apresentados separadamente na Tabela 2.3.

De acordo com a Tabela 2.3, 36,7% das heurísticas utilizadas estão relacionadas aos métodos de busca local, aleatória e escalada, e os 63,7% restantes estão relacionados a algoritmos específicos que não se enquadram nos três primeiros, por exemplo multi-início (*multi-start*) (LAMIRI; GRIMAUD; XIE, 2009) ou não definido, chamando o método apenas

como uma heurística. Este fato é parcialmente explicado pela natureza do método de busca heurística que explora uma característica específica do problema. Por esse meio, é possível utilizar um método genérico, por exemplo *Hill Climb* (escalada) (RASKA; ULRYCH, 2015), mas a aplicação da heurística infere em adaptar a busca do algoritmo pela característica específica do problema estudado.

Relacionado à metaheurística, 42,7% é derivado do algoritmo genético com denominações diferentes, por exemplo, evolução diferencial, evolução diferencial caótica, algoritmo genético, algoritmo evolucionário, NSGA II, etc. Estes são principalmente métodos de busca populacional com alguma modificação dos conceitos do algoritmo genético para o indivíduo, gene, população, crossover, etc. A soma das três metaheurísticas mais utilizadas seguintes (recozimento simulado - *simulated annealing*, *tabu* e *variable neighborhood search* -VNS, ou busca em vizinhança variável) representam 18,3% do total. De todos os 718 métodos de OvS aplicados, as 343 metaheurísticas representam 47,8% em relação ao total, e as 146 evolutivas representam 20,3%. A linha "Outros" representa 134 métodos (39,1% das metaheurísticas) combinando métodos como o algoritmo imunológico artificial, busca por dispersão, GRASP, enxame de partículas, colônia de formigas, etc.

Os "modelos substitutos" representam a segunda classe de método de OvS mais utilizado, correspondendo a 153 métodos utilizados, correspondendo 21,3% do total. O método substituto mais utilizado foi o Desenho de Experimentos (DOE) (25,5% dos métodos substitutos) seguido pelo método de "superfície de resposta" (15,7%). O método DOE em si fornece, em geral, uma regressão linear ou não, que pode não fornecer uma boa solução sozinha, mas é beneficiada de outros métodos de otimização, como a superfície de resposta. Por este meio, todos os métodos substitutos utilizam em alguma fase um metamodelo a ser otimizado, e em alguns casos, os artigos deixam claro ou não o uso deles. A linha DOE refere-se a todos os arranjos encontrados, como hipercubo, fatorial completo, Taguchi, design robusto, Hiper Cubo Latino, etc. A regressão está relacionada a métodos mais específicos, por exemplo equações de primeira ou ordens mais elevadas. Os métodos de krigagem estão relacionados a todos que usam a mesma base, por exemplo, krigagem de metamodelo, krigagem retificada, estudantizada, etc.

Para OvS, o ambiente comum apresentado nos artigos é de utilização de computador que usa um único algoritmo sequencial de instruções na forma de software proprietário ou algoritmo para gerar o modelo de SED e avaliar o espaço de busca. Em 36 estudos (9,9% do total) foram encontrados a implementação de problemas específicos que utilizavam máquinas únicas e instruções paralelas ou múltiplas máquinas com instruções distribuídas. Dessa forma,

foram encontrados dois tipos de paralelismo, relacionados ao uso de apenas CPU (Unidade Central de Processamento) em máquinas simples ou distribuídas ou algoritmos híbridos que utilizam tanto CPU quanto GPU (Unidade de Processamento Gráfico) para paralelizar o processamento de alguma instrução na simulação ou otimização.

Dos 36 trabalhos originais, 28 (77,8%) utilizaram a paralelização com a CPU e 8 (22,2%) utilizaram a GPU junto com a CPU, com uma publicação em 2010 (PARK; FISHWICK, 2010), apresentando isso como uma tendência para o futuro da OvS. A paralelização da simulação sozinha não caracteriza como um método de otimização, somente se for possível avaliar todo o espaço de busca. Para isso, é possível combinar outros métodos (por exemplo, heurísticas e metaheurísticas) para buscar uma boa solução em um período de tempo viável (COSTA et al., 2015; ESSAID et al., 2018; FUJIMOTO et al., 2017; MOKHTARI; SALMASNIA, 2015; SAILER et al., 2013; UHLIG; ROSE, 2011).

O quinto critério refere-se ao uso de programas de otimização proprietários, por exemplo OptQuest® e SimRunner®. O uso de um software de otimização proprietário, do ponto de vista acadêmico, possui limitações que comprometem o desenvolvimento e teste de novos métodos de otimização que podem apresentar contribuições para a literatura arbitrada. Esse é um dos motivos que contribuíram para o reduzido número de artigos (27 artigos) que utilizaram software de otimização proprietário como principal software de otimização ou um ponto de comparação para se relacionar com outros métodos desenvolvidos.

Os métodos relacionados a Monte Carlo e Baseados em Gradiente representam 6,8% do total (25 vezes utilizados). A categoria “Outros” refere-se a métodos que foram usados menos de cinco vezes (74 artigos, 20,3% do total), por exemplo, redução de modelo, árvore de decisão, clonagem, fuzzy, etc. O pequeno número de publicações apresentados na categoria “Outros” pode ser um sinal para a necessidade de desenvolvimento desses métodos de otimização, de acordo com o sucesso da implementação apresentada nos artigos. A Tabela 2.4 apresenta os softwares utilizados e as variáveis de OvS consideradas nos artigos.

Tabela 2.4 - Software e variáveis consideradas

			Total e % do total		
***Software*/Linguagem de programação**	**Artigos**	**Congressos**	**Total**	**%**	**Cumulativo %**
Arena®	46	21	67	13,1	13,1
Matlab®	38	16	54	10,6	23,7
Não especificado	35	18	53	10,4	34,1
C++	17	16	33	6,5	40,6
VBA®	17	06	23	4,5	45,1
Excel®	11	12	23	4,5	49,6
OptQuest®	13	11	24	4,7	54,3
Java	11	8	19	3,7	58,0
IBM CPLEX®	11	3	14	2,7	60,8
Promodel®	8	4	12	2,4	63,1
Outros	86	102	188	36,9	100,0
Medida dos resultados					
Minimização do custo	108	96	204	41,1	41,1
Maximização da produção	74	68	142	28,6	69,8
Redução do tempo de busca	31	26	57	11,5	81,3
"Benchmark"	19	11	30	6,0	87,3
Outros	36	27	63	12,7	100,0

De acordo com a Tabela 2.4, o critério "software" apresenta os programas utilizados para a OvS. O software mais utilizado é o modelador de SED Arena®, utilizado em 67 trabalhos (13,1% do total). O segundo software mais citado é o Matlab®, que pode ser utilizado para modelagem ou otimização, em 54 trabalhos (10,6%). O terceiro critério "Não especificado" representa os artigos que utilizaram algum tipo de linguagem de programação, mas não especifica qual, citada em 53 artigos (10,4%). Os três *softwares* mais utilizados trazem à tona o problema relacionado a todos os estudos de OvS que é a geração do modelo computacional em SED e a chamada recursiva para a avaliação dos resultados por um método de otimização e os parâmetros para chamar os novos cenários, necessitando assim de um modelador e um algoritmo de otimização.

Por essa razão foi comum a presença de mais de um software em todos os artigos que se utilizaram de um software proprietário para a modelagem da SED, a presença também de algum tipo de linguagem de programação, a exemplo do software Arena® estar associado em sua maioria com a linguagem de programação VBA® (SOYKAN; RABADI, 2016; UPADHYAY; ASKARI-NASAB, 2018).

Analisando a Tabela 2.4, é possível inferir que não há consenso nas comunidades acadêmicas e profissionais para definir um *framework* capaz de unir as fases de modelagem e otimização nos projetos de OvS. Considerando que não é uma prática comum usar mais de um modelador de SED, pode-se dizer que pelo menos 133 artigos, dos 363 analisados, usaram um

programa comercial para modelar a SED, representando cerca de 36,6% de todos os estudos, e a maioria dos estudos usou uma linguagem de programação, exceto os artigos que usaram apenas programas de otimização comercial (por exemplo, OptQuest® e SimRunner®). A variedade e dispersão para possíveis combinações de métodos de OvS e "bancadas de teste" empregadas, dificultam a replicação dos estudos ou a comparação dos resultados de um método de otimização. Como esforço no sentido de desenvolvimento de um *framework* em OvS capaz de criar um modelo de SED e um ambiente aberto de testes para diferentes otimizações, podem ser citados os trabalhos de Freitag e Hildebrandt (2016) e Hildebrandt, Goswami e Freitag (2015).

Para responder a PP2 "Quais métodos de otimização e software de implementação foram os mais utilizados?" é possível analisar a Tabela 2.4 e os comentários, que resumem a fase em que o problema é traduzido para um modelo computacional. Em quase metade dos casos foi utilizado um modelador comercial de SED e, para a otimização, foi utilizada uma linguagem de programação, implementando, na maior parte dos casos, uma análise de modelos metaheurísticos ou substitutos.

A última categoria na Tabela 2.4 está relacionada aos principais indicadores de desempenho que constituíram as variáveis para a função objetivo e as restrições que limitam o espaço de busca, separadas em quatro critérios. Os estudos de *benchmark* estão relacionados a trabalhos teóricos com objetos de estudo que utilizaram dados da literatura (livros e artigos especializados) ou compararam os resultados do autor com programas de otimização comercial, responsáveis por 30 variáveis, 6,0% do total de estudos. A "maximização da produção" são os objetivos relacionados aos tempos de processo que influenciam os parâmetros de produção, por exemplo, comprimento da fila, taxa de produção, tempo de fluxo produtivo, tempo de espera, *lead time*, produção do produto, *makespan*, etc., correspondentes a 142 variáveis (28,6%).

A "minimização do custo" relaciona as variáveis que influenciam diretamente os custos de produção, como folha de pagamento, receita, desempenho, eficiência, lucro líquido, capital, recursos, redução de custos, WIP (*Work in Process*), tamanho do lote, nível de estoque, tamanho do *buffer*, tamanho do lote de reposição, etc., mostrando em 204 artigos, ou 41,1% do total. A "Redução do tempo de busca" está relacionada ao tempo ou número de iterações necessárias para encontrar boas soluções, presentes em 57 artigos, ou 11,5% do total. O critério "Outros" refere-se a outras medidas de desempenho, a exemplo de estatísticas coletadas entre resultados como o Erro Médio Absoluto e o Erro Médio Quadrático (em inglês *Mean Average Error* - MAE e *Mean Squared Error* - MSE).

Observou-se que a medição dos resultados foi feita para correlacionar pelo menos um critério de maximização e minimização de custo sobre os objetivos e/ou restrições. Isto é esperado para a formulação do problema de otimização com uma solução espacial definida e finita, com variáveis relacionadas inversas que em algum ponto têm uma região que maximiza ou minimiza a resposta do problema. Na maioria dos artigos não ficou claro se o melhor resultado apresentado foi implementado nos sistemas reais, nem quão longe foi do provável ótimo global. Por meio das análises realizadas, pode-se constatar que, parte dessa questão pode ser explicada pelo fato de que um projeto de OvS por si mesmo já demanda tempo para sua execução, estando os resultados de sua implantação em outro projeto. Esta questão pode ser melhor explorada em trabalhos futuros.

Considerando a PP3 "Como os resultados foram medidos?" é possível inferir que a mensuração dos projetos de OvS está relacionada ao propósito inicial que estimula o desenvolvimento dos mesmos. Para um projeto geral de OvS sobre EP, as razões estavam relacionadas à necessidade de avaliação em múltiplos cenários que influenciam a maneira como a organização trabalha e a sua lucratividade, em que uma simulação manual seria proibitiva para avaliar a totalidade das possíveis soluções. Portanto, a medida de avaliação de qualidade da solução é relacionada a maximização dos objetivos e o tempo necessário para a otimização chegar nela.

Segundo Chwif, Paul e Barretto (2006), os métodos e procedimentos de otimização aplicados a SED podem ser classificados em quatro categorias: busca baseada em gradientes, aproximações estocásticas, metodologia de superfície de resposta e métodos de busca heurística. Essa categorização foi feita em 1999 e não era o objetivo do artigo fazer uma definição mais precisa dos métodos de OvS. Juan et al. (2015) considerou o ranking e seleção, métodos de busca caixa-preta, meta-modelo, métodos baseados em gradiente, caminho de amostra e restrições estocásticas e multi-objetivo. A Figura 2.9 resume os achados relacionados às metodologias aplicadas a OvS nos problemas de EP que ampliam a classificação proposta dos autores supracitados e também relacionados na Seção 2.3.

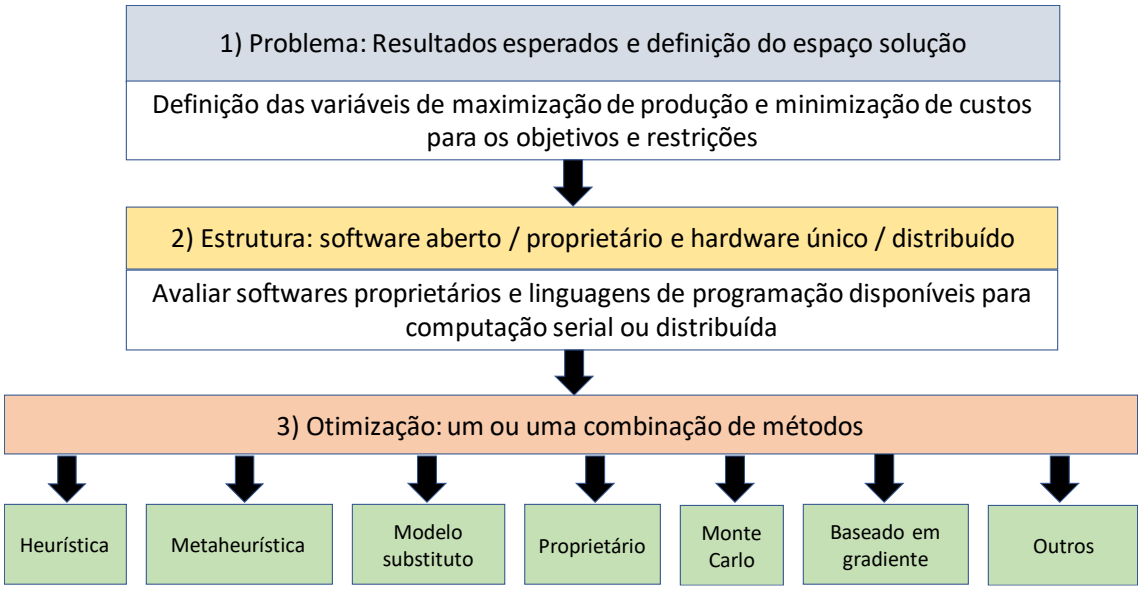

Figura 2.9 - Resumo dos métodos de OvS aplicado a EP.
Fonte: O autor

A Figura 2.9 ilustra as melhores práticas para OvS em EP encontradas no presente estudo para ajudar no planejamento das etapas de projetos dessa área, com base nos 363 artigos avaliados. O primeiro passo é a definição do problema e as questões que serão respondidas pela OvS. Se o espaço de pesquisa ou o problema puderem ser limitados em um pequeno número de cenários, métodos intuitivos podem ser usados para determinar o número de mudanças manuais necessárias no modelo de simulação. Dessa forma, escapam do escopo do presente trabalho, que avaliou problemas que têm um espaço de pesquisa proibitivo de ser feito manualmente, mas apenas com a ajuda de um método de otimização computacional. Após a avaliação do problema e a necessidade de um método de busca automática/recursiva, é possível avaliar o software necessário para o modelo e otimização computacional.

O segundo passo consiste em avaliar os recursos disponíveis referentes a tempo e conhecimento para empregar na construção e busca da solução ideal. Isto constitui uma questão importante, porque os sistemas de simulação mais complexos envolvendo simulação distribuída/paralela exigem mais tempo e investimento em equipamento e não são uma garantia para encontrar uma solução melhor do que uma programação sequencial, mas aumenta a probabilidade para tal efeito. O último passo é determinar o(s) método(s) de otimização aplicado(s) ao modelo de simulação. A seleção do método pode ser relacionada às descobertas de trabalhos anteriores ou pela implementação e comparação de diferentes métodos, de acordo com os recursos disponíveis. Neste ponto, não há um consenso sobre qual método é mais adequado para resolver o problema, dependendo do conhecimento prévio e da experiência da equipe de simulação.

2.6.3 Origem da pesquisa

A "origem da pesquisa" sumariza os dados coletados nos 363 artigos que remetem ao contexto dos autores e onde produziram os artigos relacionados a OvS em EP, para auxiliar na resposta da PP4. A escolha de analisar os autores envolvidos é devida ao interesse do presente estudo em descobrir se existe uma correlação entre os autores e suas diferentes nacionalidades para construir uma comunidade de pesquisa na área de OvS, e para identificação (ou não) de centros de excelência na área. A Tabela 2.5 apresenta os resultados encontrados.

Tabela 2.5 - Número de publicações de acordo com o nome do autor, filiação e nacionalidade

TOP 10	**Nome - Número de publicações**	
Pesquisador	**Publicações em Periódicos**	**Publicações em Congressos**
1	Jack P.C. Kleijnen – 6	Amos H.C. Ng – 4
2	A. Azadeh – 5	Hongwei Ding – 3
3	B. Naderi – 3	Lars Mönch – 3
4	Berna Dengiz – 3	Lyes Benyoucef – 3
5	Feng Yang – 3	Torsten Hildebrandt – 3
6	Wim C.M. van Beers – 3	Xiaolan Xie – 3
7	Wout Dullaert – 3	Alexander Pacholik – 2
8	Amos H.C. Ng – 3	Alexandre Ferreira de Pinho – 2
9	Richard F. Hartl – 2	Andrés Muñoz-Villamizar – 2
10	Christian Almeder – 2	Anna Persson – 2
Filiação		
1	Amirkabir Univ. of Technology – 9	Dresden University of Technology – 6
2	University of Tehran – 6	Purdue University – 5
3	University of Vienna – 5	University of Paderborn – 4
4	Louisiana State University – 5	University of Skövde – 4
5	Tilburg University – 4	Ilmenau Technical University – 3
6	University of Antwerp – 4	Nanyang Technological University – 3
7	Baskent University – 3	Northeastern University – 3
8	Ghent University – 3	Tongji University – 3
9	Islamic Azad University – 3	University of Hagen – 3
10	Shanghai Jiao Tong University – 3	Durham University – 2
Nacionalidade		
1	United States – 42	United States – 38
2	Iran – 25	Germany – 26
3	China – 16	China – 17
4	Germany – 12	Italy – 10
5	Brazil – 9	United Kingdom – 8
6	Canada – 9	France – 8
7	United Kingdom – 8	Brazil – 8
8	France – 7	Sweden – 5
9	Belgium – 6	Colombia – 4
10	Italy – 6	Ireland – 3

Nos dados coletados para construção da Tabela 2.5, do total de 789 nomes encontrados em todos os artigos, 126 aparecem mais de uma vez e foram encontrados 663 nomes sem repetição. Com as informações da Tabela 2.5 e o ranqueamento dos *Top 10* autores mais produtivos nessa área, a diferença do autor mais produtivo para os outros varia entre 1 e 4

artigos, mostrando um panorama em que os autores não têm uma produção contínua apenas no campo da OvS aplicado a problemas de EP, com um domínio esparso desse conhecimento. Esse efeito exige uma pesquisa por diferentes autores para encontrar mais detalhes sobre uma metodologia de otimização específica ou problema relacionado em que se deseja estudar.

A segunda informação da Tabela 2.5 está relacionada com a filiação dos autores. Com um entendimento semelhante da conclusão anterior, as universidades onde os estudos foram realizados não apresentam um número expressivo de publicações que justifiquem um *cluster* ou centro de pesquisa referência, relacionando exclusivamente a temática de OvS em EP. As últimas informações da Tabela 2.5 juntam os artigos de acordo com a filiação dos países dos autores. Neste ponto é possível fazer uma segregação de dados mais relevante, conforme a Figura 2.10.

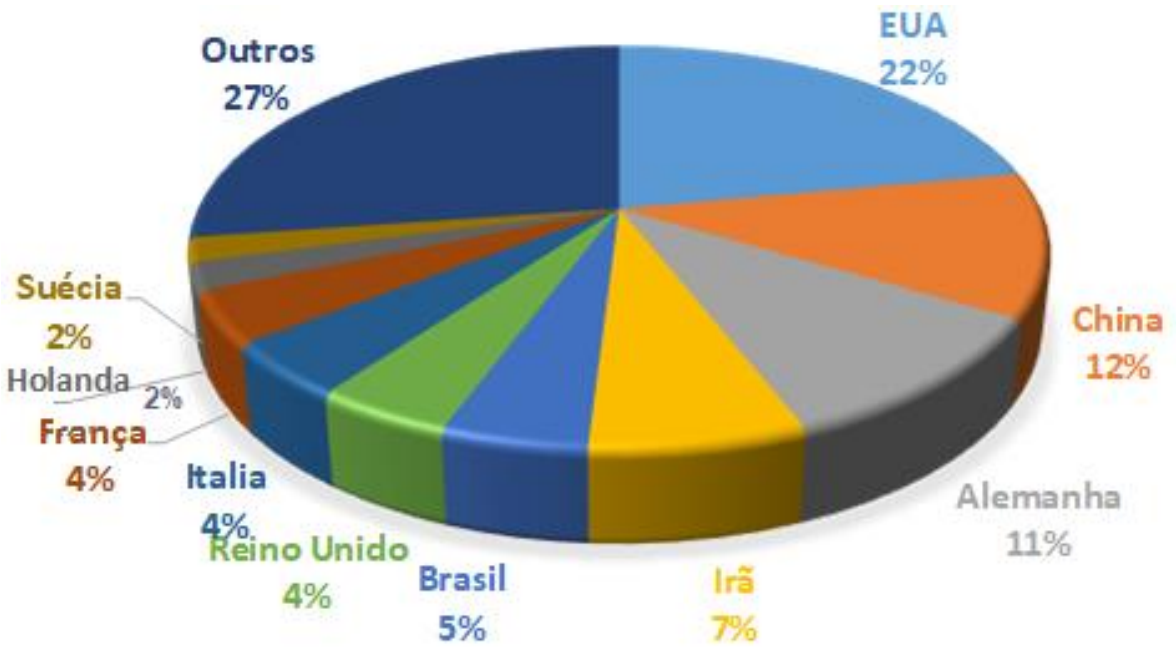

Figura 2.10 - Os 10 países que mais pesquisam a OvS em EP
Fonte: O autor

De acordo com o gráfico apresentado na Figura 2.10, é possível inferir que os EUA, China, Alemanha, Irã, Brasil, Reino Unido, Itália, França, Holanda e Suécia são os 10 principais países que concentram os autores mais produtivos, mas como segunda conclusão da informação, mesmo nesses países, o desenvolvimento da OvS em EP não tem um centro de referência específico. Outra informação relevante extraída dos artigos é como os projetos de otimização foram desenvolvidos, com a colaboração de múltiplos centros de pesquisa de

diferentes universidades, instituições do setor privado e países onde foi realizada a pesquisa. A Tabela 2.6 apresenta os dados coletados para a equipe dos autores dos artigos.

Tabela 2. 6 - Diferentes origens dos autores

	Total e percentagem do total			
Filiação dos autores	**Publicações em Periódicos**	**Publicações em Congressos**	**%**	**Cumulativo %**
Mais de uma universidade	88	20	29,8	29,8
Junto com outra instituição pública ou privada	18	51	19,0	48,8
Mais de um país de origem	31	14	12,4	61,2
Mesmo país de origem	65	76	38,8	100,0

Os dados da Tabela 2.6 estão relacionados às diferentes origens e configurações da equipe dos autores de cada artigo. Para a primeira informação é possível inferir que em 29,8% dos trabalhos, envolveu mais de uma instituição universitária. A segunda informação está relacionada ao envolvimento de instituições privadas e públicas, diferente de uma organização acadêmica, responsável por 19,0%. Esta questão assina a interação entre acadêmicos e o setor econômico ou instituições governamentais. O terceiro dado correlaciona autores que possuam filiação institucional em diferentes países, para 12,4% dos trabalhos. Para o restante das publicações, 38,8% dos autores são da mesma instituição acadêmica. A Tabela 2.7 relacionou os 10 principais periódicos de publicação e os congressos encontrados para a OvS em EP.

Tabela 2.7 - Top 10 locais de publicação para OvS em EP.

Rank	**Periódicos**	**Publicações**	**%**	**Cum%**
1	European Journal of Operational Research	22	6,1	6,1
2	Computers and Industrial Engineering	17	4,7	10,7
3	International Journal of Production Economics	12	3,3	14,0
4	Simulation: Transactions of the Society for Modeling and Simulation International	12	3,3	17,4
5	Computers and Operations Research	10	2,8	20,1
6	Simulation Modelling Practice and Theory	10	2,8	22,9
7	International Journal of Advanced Manufacturing Technology	7	1,9	24,8
8	Journal of Manufacturing Systems	6	1,7	26,4
9	Applied Soft Computing	3	0,8	27,3
10	Engineering Applications of Artificial Intelligence	3	0,8	28,1

Rank	Congressos			
1	Winter Simulation Conference	84	23,1	23,1
2	International Conference on Automation Science and Engineering	3	0,8	24,0
3	Conference on Manufacturing Modelling, Management and Control	2	0,6	24,5
4	European Conference on Modelling and Simulation	2	0,6	25,1
5	European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering	2	0,6	25,6
6	International Conference on Industrial Engineering and Engineering Management	2	0,6	26,2
7	International Conference on Service Systems and Service Management	2	0,6	26,7
8	International ICST Conference on Simulation Tools and Techniques	2	0,6	27,3
9	SIGSIM-PADS	2	0,6	27,8
10	World Congress on Intelligent Control and Automation	2	0,6	28,4

De acordo com a Tabela 2.7, os 10 periódicos apresentados representam 102 artigos publicados em OvS para EP, com 41,4% do total de publicações. Para os trabalhos de congressos e conferências, o *Winter Simulation Conference* (WSC) representa cerca de 23,1% de todos os artigos, com uma diferença para o segundo lugar de 23%, de um total de 45 congressos e conferências, mostrando que o WSC é uma referência para as pesquisas e práticas sobre OvS aplicadas a EP. A Figura 2.11 ilustra as publicações de OvS em EP ao longo dos 27 anos (1991-2018) considerados para a presente pesquisa.

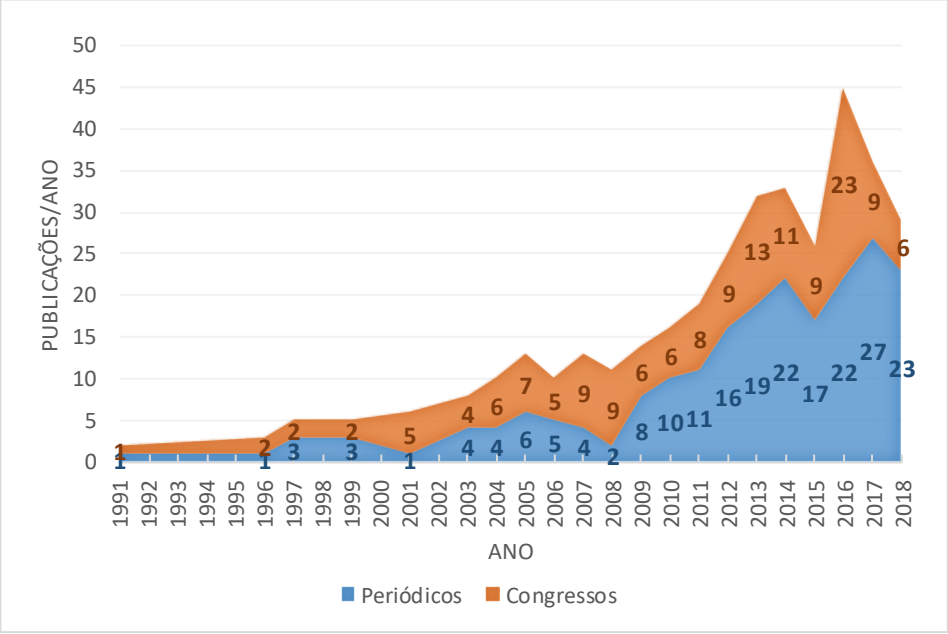

Figura 2.11 - Publicações de OvS em EP.

De acordo com a Figura 2.11, é possível inferir que nos 17 anos iniciais, de 1991 a 2008, existe um período de crescimento e amadurecimento dos conceitos e aplicações da OvS em EP. Isso pode ser parcialmente explicado pela evolução da Teoria de Filas e acessibilidade a computadores e *software* de SED. De 2008 a 2018, existe uma tendência de crescimento, porém com um declínio em 2015 sem estar claro os motivos. Uma característica observada, que não é tratada no presente trabalho, é que no mesmo período a modelagem baseada em agentes estava crescendo. Isso pode ser um sinal para a mudança de direção dos pesquisadores e praticantes para estudar sistemas híbridos ou diferentes tipos de simulação sem descartar a metodologia necessária para desenvolver um projeto de simulação, mas é muito cedo para inferir com precisão essa observação, que precisa de um período maior de tempo e dados para ser conclusivo.

Finalizando com a PP4 "Qual autor, universidade, ano de publicação e periódico foram encontrados e que compõem os centros de pesquisa de referência?" Não é possível inferir com as informações nas Tabelas 2.5 a 2.7 e na Figura 2.11, a existência de um centro de pesquisa de referência que concentra as publicações de OvS em EP, além disso, os dados coletados inferem uma orientação dispersa para a origem dos trabalhos publicados.

2.6.4 Direcionamento da pesquisa em OvS

De acordo com os trabalhos avaliados no presente estudo, três métodos de otimização mostraram bons resultados no auxílio à obtenção respostas boas e ainda estão em desenvolvimento. Os dois primeiros estão relacionados ao uso de metaheurísticas e algoritmos de otimização por aprendizado de máquina, e o último método com paralelização via hardware (DE SOUSA JUNIOR et al., 2019).

Para aplicar técnicas de busca em problemas NP-difícil, com um espaço de solução disperso, metaheurísticas podem ter um bom desempenho em termos de encontrar soluções ótimas locais ou quase globais, em um tempo razoável. Se a simulação a eventos discretos for necessária, métodos substitutos de modelos (ou metamodelos) podem ser usados em vez da simulação real. Os metamodelos são, em geral, uma representação matemática que dá um resultado semelhante à simulação real, em um tempo menor que o necessário para executar a simulação. Por exemplo, é possível citar o uso do método de aprendizado de máquina por Árvores de Decisão e a metaheurística Busca Tabu para regras de despacho de produção (SHAHZAD; MEBARKI, 2016).

A mesma ideia é encontrada em artigos mais recentes, usando os métodos de otimização por “sistemas imunológicos artificiais” e “algoritmo genético” em um sistema de manuseio de materiais (LEUNG; LAU, 2018), e o uso de algoritmos evolutivos NSGA-II e SPEA2 para selecionar bibliotecas ótimas de Infraestrutura em Tecnologia da Informação (RUIZ et al., 2018) e o uso do algoritmo de evolução diferencial multi-objetivo baseado em decomposição (MODE/D) comparado com o NSGA-II para o problema de reposição de estoque (AVCI; SELIM, 2018).

O terceiro método é o uso de paralelismo via hardware. Junto com o desenvolvimento de algoritmos de otimização, os computadores modernos aumentaram seu poder computacional, em especial a capacidade de processar mais de uma instrução por vez com o advento da popularização de processadores com mais de um núcleo, nos quais a simulação de eventos discretos pode ser beneficiada (FUJIMOTO et al., 2017; JAFER; LIU; WAINER, 2013). Uma pesquisa recente utilizou o NSGA-II e o paralelismo em projetos de construção de pontes, sendo o tempo, para encontrar a solução, uma das características principais do problema (SALIMI; MAWLANA; HAMMAD, 2018).

Os três métodos citados - metaheurística, metamodelo por aprendizado de máquina e processamento paralelo - têm em comum a necessidade de encontrar boas soluções em baixo tempo computacional. Esta questão reflete a crescente necessidade de processar uma grande quantidade de dados estocásticos e se beneficiar dos desenvolvimentos em algoritmos de otimização e paralelização via hardware. Essa constatação de boas práticas de OvS é parte da justificativa dos métodos que foram utilizados para o presente trabalho.

3. MÉTODO DE PESQUISA

No presente capítulo é apresentado o referencial teórico metodológico levado em consideração na elaboração da presente tese, visando delimitar os resultados pretendidos com a definição do método científico empregado. São apresentados a classificação da pesquisa, o método de pesquisa, etapas da pesquisa e a aplicação das etapas da pesquisa.

3.1 Classificação da pesquisa

De acordo com Turrioni e Mello (2012), a classificação das pesquisas científicas realizadas na área de conhecimento referente à Engenharia de Produção, possuem três dimensões básicas referentes a sua natureza, objetivos pretendidos e a abordagem para atender os objetivos pretendidos.

No que se refere a "natureza da pesquisa", as pesquisas podem ser categorizadas como "básicas", quando fazendo referência ao desenvolvimento conceitual sem a preocupação de imediata aplicação, ou "aplicada", caso contrário (TURRIONI e MELLO, 2012). O presente trabalho pode ser avaliado como "aplicado", pois há o interesse de que o mesmo seja empregado em objetos de estudos que representam problemas encontrados no cotidiano da área de Engenharia de Produção.

Na dimensão de pesquisa "objetivos", os mesmos podem ser classificados como "exploratórios" quando visa tornar explícito ou construir hipóteses para um problema; "descritivos" quando tem por objetivo descrever as características de determinado assunto; "explicativos" ao se determinar os fatores que contribuem para a ocorrência do fenômeno desejado; e "normativos" ao se tentar ou encontrar uma solução ótima para novas definições de problemas definidos ou para comparar várias estratégias e políticas em relação a um problema foco do estudo (TURRIONI e MELLO, 2012). Para o presente trabalho, o mesmo pode ser classificado como normativo, pois tem o interesse de testar várias políticas de otimização para objetos de estudo específicos.

Na dimensão "abordagem", é definido que para se conduzir a pesquisa e comprovar o atendimento dos objetivos, é necessário adotar métodos quantitativos, qualitativos ou uma combinação dos dois para a geração e análise de dados numéricos ou subjetivos (TURRIONI e MELLO, 2012). Dentre as abordagens quantitativas mais importantes, de acordo com Miguel et al. (2012) e Turrioni e Mello (2012), pode-se relacionar: os experimentos, a pesquisa levantamento (*survey*) e a modelagem e simulação. Considerando os métodos qualitativos pode-se citar: estudo de caso, pesquisa-ação, e *soft systems methodology*. O método que melhor

define as variáveis envolvidas e os objetivos da presente pesquisa é a Modelagem e Simulação. A Figura 3.1 apresenta o resumo das classificações de pesquisa em Engenharia de Produção, assim como em negrito as abordagens adotadas pela presente tese.

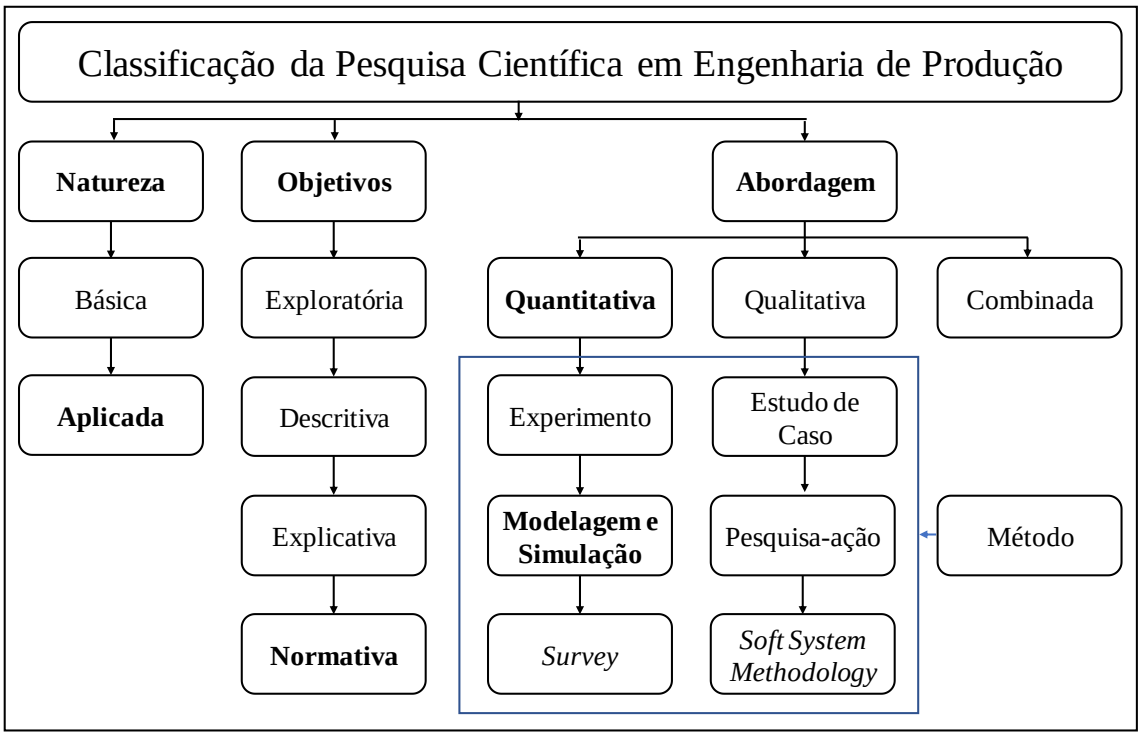

Figura 3.1 - Classificação da pesquisa científica em Engenharia de Produção
Fonte: Adaptado de Miguel et al. (2012)

A partir da Figura 3.1 é possível inferir a classificação da presente tese como sendo uma abordagem por Modelagem e Simulação com a predominância de variáveis quantitativas para geração e análise de dados, no intuito de se gerar um estudo normativo aplicado. Na Seção 3.3 é melhor definido o método de Modelagem e Simulação para o estudo.

3.2 Método de pesquisa

A metodologia científica na área da Pesquisa Operacional para SED teve o trabalho descritivo seminal desenvolvido por Mitroff et al. (1974). Em seu modelo, os autores propõem quatro fases e as ações necessárias para se passar de uma fase para outra. O modelo está apresentado na Figura 3.2.

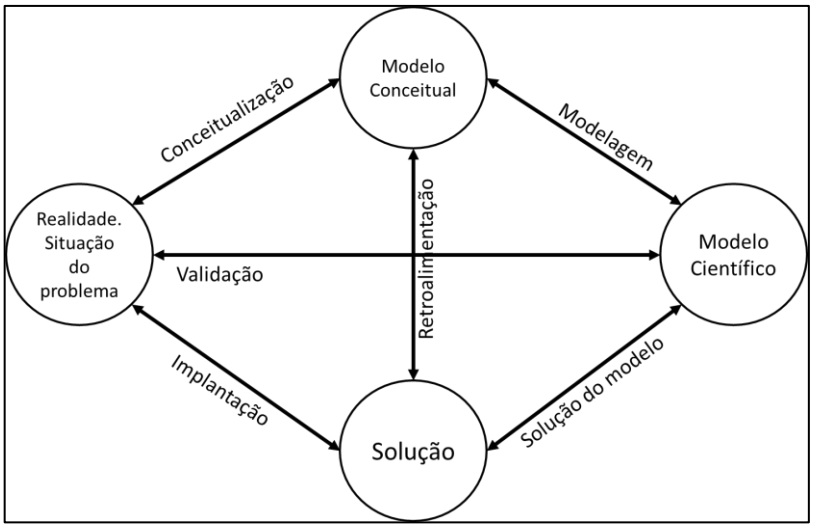

Figura 3.2 - Visão do sistema de resolução de problemas
Fonte: Adaptado de Mitroff et al. (1974)

Analisando o modelo de Mitroff et al. (2002), Bertrand e Fransoo (2002) propõem quatro tipos de pesquisa quantitativa com base em modelos:

- Empírica descritiva: nesse tipo de trabalho, o pesquisador segue o ciclo de: conceitualização, modelagem e validação;
- Empírica normativa: é a forma de trabalho contendo conceitualização, modelagem, solução do modelo e implantação;
- Axiomática descritiva: apenas a ação modelagem, indo da fase de modelo conceitual para modelo científico;
- Axiomática normativa: utiliza das ações de modelagem e solução do modelo.

Com base nas classificações de Bertrand e Fransoo (2002), Miguel et al. (2012) definem os trabalhos axiomáticos normativos para modelagem e simulação como:

> "Na pesquisa axiomática normativa, novos modelos de otimização ou variações de modelos existentes podem ser propostos para um problema idealizado, como problemas de dimensionamento de lotes e de roteamento de veículos dos exemplos citados, utilizando métodos de solução conhecidos da literatura para resolvê-los. Também podem ser desenvolvidas pesquisas que estudam modelos de otimização conhecidos para problema idealizado, mas que propõem novos métodos para resolver esses modelos ou variações de técnicas de solução já existentes, mas que produzem melhores resultados. Nesses casos, além de conhecimento de teoria e otimização matemática, também é necessário conhecimento nas áreas numéricas e ciências da computação" (Miguel et al., 2012, p 180).

A partir da definição anterior, é possível inferir que esta tese adota o método axiomático normativo para sua elaboração, pois não há a pretensão de se criar um modelo novo a partir da

observação de um sistema real e transformá-lo em modelo conceitual, mas de transformar modelos conceituais já existentes em modelos científicos, de tal maneira que possibilite a otimização de modelos de SED com metaheurística, metamodelagem e paralelismo.

3.3 Etapas da pesquisa

A presente tese adotou a abordagem proposta por Bryman (2003). De acordo com o referido autor, as etapas de uma pesquisa quantitativa podem ser ilustradas pelas informações da Figura 3.3.

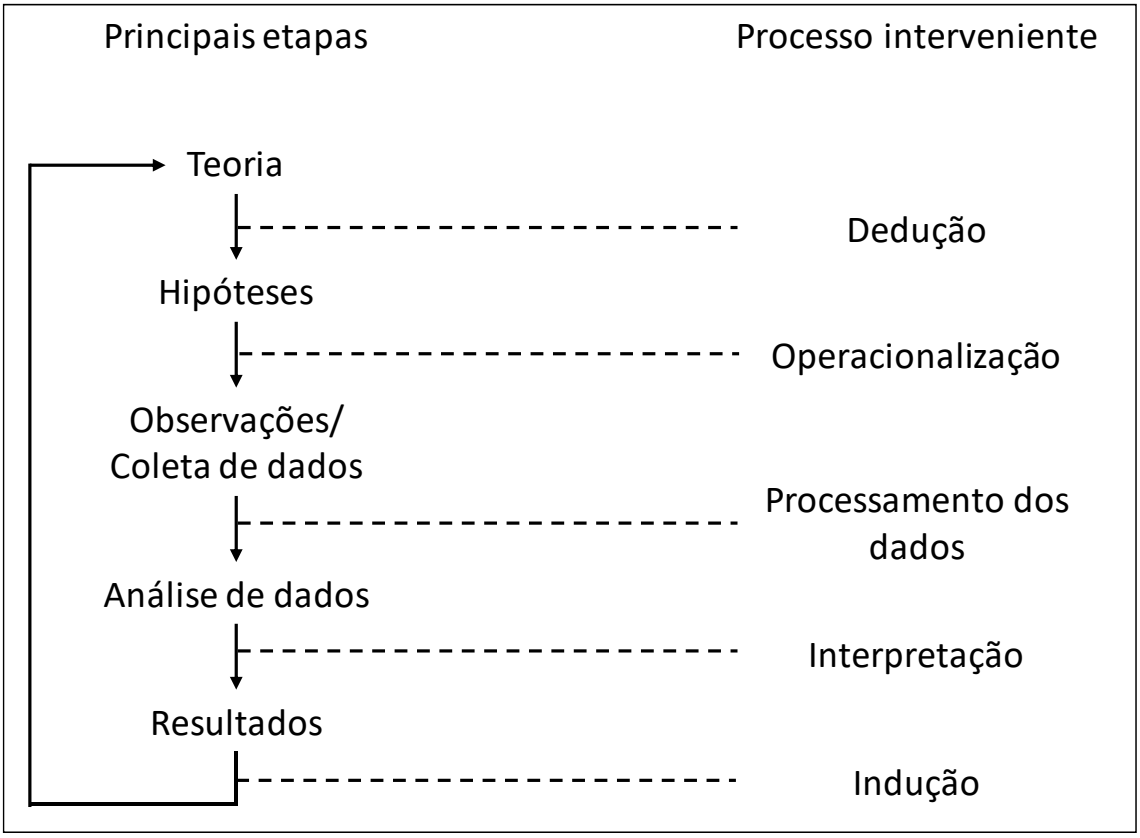

Figura 3.3 - A estrutura lógica do processo de pesquisa quantitativa
Fonte: Adaptado de Bryman (2003)

Analisando a Figura 3.3, Bryman (2003) afirma que o processo inicial é de definição de um problema que englobe as variáveis que melhor representam os sistemas envolvidos. Para esse processo de definição, segundo o autor supracitado, grande quantidade de tempo é necessária para que a literatura técnica da área seja consultada, a fim de se conhecer como problemas parecidos já foram definidos, o que foi utilizado para resolver os mesmos e quais os resultados esperados em situações parecida.

A partir da literatura selecionada, um processo de dedução deve ser empregado para que as hipóteses principais do trabalho sejam relacionadas para se determinar quais os tipos de respostas serão esperados ao seu término. Com a definição das hipóteses, um processo de coleta, geração, análise e síntese de dados é estruturado e realizado para que as hipóteses iniciais sejam confirmadas total ou parcialmente com ressalvas, devido ao fato de que informações foram sendo angariadas durante a pesquisa e que não foram consideradas no início da mesma.

Desta forma, a presente tese adotou a abordagem proposta por Bryman (2003) com a definição do problema e os referenciais técnicos de trabalhos semelhantes. A elaboração do Capitulo 2, com aplicação do método de revisão sistemática da literatura, proporcionou a definição de constatações para se determinar quais os tipos de problemas mais aptos para se empregar a otimização via simulação e quais as metodologias de otimização foram aplicadas com sucesso e podem ser mais aptas para serem empregadas no presente trabalho, gerando assim as fases de Teoria, Dedução e Hipóteses propostas por Bryman (2003).

A partir da determinação das melhores práticas para solução de problemas da ordem de OvS, o presente Capítulo 3 definiu o construto a ser seguido para a elaboração do método. No Capítulo 4 foram definidos os objetos de estudo e quais requisitos principais deveriam estar presentes para a seleção dos *softwares*, de maneira tal que fosse possível a operacionalização da proposta de tese com a geração de uma modelagem computacional. A partir da modelagem, deve ser possível gerar e testar um novo método de otimização que aplicasse técnicas de: metaheurística, aprendizagem de máquina e paralelismo em um mesmo ambiente, além da definição dos critérios de comparação entre os métodos testados.

A partir do método proposto no Capítulo 4, o ambiente de programação foi implementado (fase de Operacionalização) e dados foram gerados e apresentados no Capítulo 5, com sua análise e confirmação das hipóteses iniciais da tese, de maneira tal que o método proposto se mostrou eficiente para os objetos de estudo apresentados (fases de Coleta de dados e Processamento dos dados). A Figura 3.4 ilustra um fluxograma com os passos propostos para a programação e geração dos dados desejados.

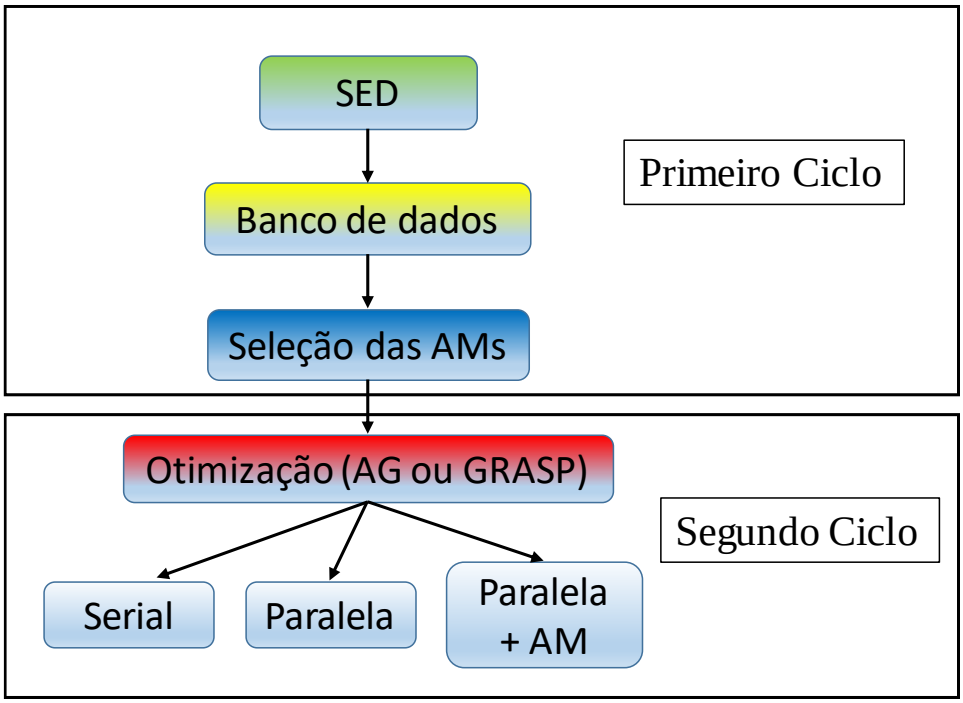

Figura 3.4 - Ciclos de programação para geração e análise de dados

De acordo com a Figura 3.4, a operacionalização do método de otimização proposto é dividida em dois Ciclos. O primeiro ciclo se refere a seleção do ambiente de modelagem computacional para a Simulação a Eventos Discretos, em que os Objetos de Estudo são modelados e utilizados como fonte para gerar as bases de dados (*data frame*) e testar os métodos de Aprendizagem de Máquina. A partir do treino e teste dos métodos de AM, são selecionados os melhores métodos de AM. O segundo ciclo (de acordo com a Figura 3.4) da otimização utilizando de metaheurísticas populacionais (GA ou GRASP) para teste dos Objetos de Estudo (OEs) utilizando de programação serial, paralela e paralela com a incorporação no algoritmo de otimização dos métodos e AM selecionados no primeiro ciclo.

No Capítulo 6 foram comparados os resultados para se avaliar a qualidade das respostas obtidas dos algoritmos de otimização em comparação aos objetivos da tese, além da apresentação de propostas de trabalhos futuros (fases de: Análise de dados, Interpretação, Resultados e Indução).

4. MÉTODO PROPOSTO

Neste Capítulo são apresentados os passos necessários para a construção e validação para o ambiente computacional proposta no Capítulo 3, com foco na implementação e análise, assim como proposto em Montevechi et al., (2010), principalmente sobre a seleção dos componentes relacionados a sua execução. Esses componentes incluem três objetos de estudo, um ambiente de SED e otimização, as técnicas de aprendizado de máquina, as metaheurísticas e métricas para avaliação dos resultados.

4.1 Objeto de estudo

Na presente pesquisa, uma abordagem indutiva foi utilizada para testar e confirmar os ganhos no uso de paralelismo, aprendizado de máquina e metaheurística em uma mesma estrutura. Como mostrado em estudos, um único objeto de estudo é possível de ser usado em tais casos (MARSCHAN-PIEKKARI; WELCH, 2004; SALAM; KHAN, 2016). O modelo de três objetos de estudo foi utilizado com múltiplos cenários de técnicas de otimização, a fim de encontrar a melhor combinação de acordo com o tempo necessário para obedecer ao critério de parada do algoritmo desenvolvido.

A seleção dos três objetos de estudo ocorreu de acordo com os dados presentes na Tabela 2.2, com a determinação dos principais problemas de OvS. Dessa forma, os três objetos de estudo apresentados são referentes a "Alocação de Recursos no Chão de Fábrica" (OE1), "Controle de Estoque" (OE2) e determinação de "Máquinas e *Buffers*" (OE3), representando de cerca de 40,0% (somadas as probabilidades) dos problemas estudados nessa área.

4.1.1 Primeiro objeto de estudo: Alocação de recursos no chão de fábrica

O primeiro objeto de estudo (OE1) é adaptado de estudos anteriores relacionados à alocação de recursos no chão de fábrica (AZADEH; ASADZADEH; TADAYOUN, 2014; AZADEH; MOTEVALI HAGHIGHI; ASADZADEH, 2014). Máquinas paralelas puxam recursos para serem processados e geram mercadorias com uma taxa estocástica e probabilidade de quebra. A equipe de manutenção repara as máquinas quebradas com uma taxa constante. Se o número de máquinas desativadas for maior que o da equipe, ele aguarda até que alguém esteja disponível. Quando a máquina quebra, a produção em processo é descartada. Todos os valores usados na simulação são mostrados na Tabela 4.1.

Tabela 4.1 - Distribuições estatísticas das variáveis do primeiro objeto de estudo

Atividade	**Distribuições de probabilidade (min)**
Tempo de processamento	Normal (média = 10, desvio padrão = 2)
Tempo médio entre falhas	Exponencial (lambda = 1/300)
Tempo para reparo	Constante = 30

A função objetivo e as restrições do primeiro objeto de estudo são apresentados nas Equações (3-6), sendo usadas para avaliar cada cenário de simulação e encontrar o lucro máximo, a partir da interação das respostas estocásticas geradas pelas funções da Tabela 4.1, de acordo com a simulação.

$$Maximizar\ FO(máquinas_i;\ pessoal_j;\ tempo_k) = \left(\sum peças\_produzidas_{maquinas_i} \times \$100\right) - \left(pessoal_j \times \$20 \times tempo_k\right) - \left(tempo_k \times \$5\right) \quad (3)$$

Sujeito a:

$$Máquinas_i = \{1;\ 2;\ \ldots;\ 40\}: \forall i \in máquinas\ \in\ \mathbb{Z} \quad (4)$$

$$Pessoal_j = \{1;\ 2;\ \ldots;\ 20\}: \forall j \in pessoal\ \in\ \mathbb{Z} \quad (5)$$

$$Tempo_k = \{30;\ 31;\ \ldots;\ 150\}: \forall k \in tempo \in\ \mathbb{Z} \quad (6)$$

As variáveis que consistem na resposta de otimização são formadas pelo número de peças total geradas pelas máquinas trabalhando em paralelo e pelo número de pessoas referentes à equipe de manutenção necessária para gerar o lucro máximo, em um período de tempo variável medido em semanas. O espaço de solução para este problema é a combinação das 96800 possíveis soluções (40 x 20 x 121). Na Figura 4.1 é apresentado a representação gráfica do OE1.

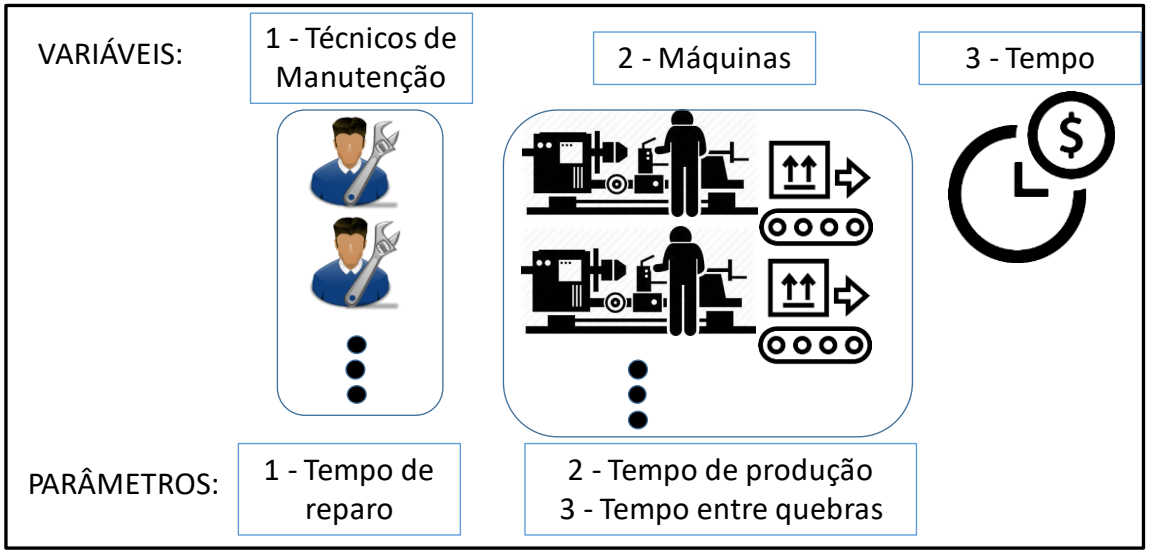

Figura 4.1 – Representação gráfica do OE1

A escolha deste problema específico foi definida pelo fato dos problemas de fabricação relacionados à alocação de recursos no chão-de-fábrica terem um grande impacto na forma como a empresa trabalha com investimentos, planejamento mestre da produção, alocação de *buffer*, regras de despacho, flexibilização da produção, etc. (GEYIK; DOSDOĞRU, 2012; PONSIGNON; MÖNCH, 2014; WANG et al., 2015; YANG; KUO; CHO, 2007). Outro motivo que justifica a utilização de técnicas de otimização para este problema é a necessidade de entorno de 22 dias para se gerar todo o espaço solução para este problema. A modelagem conceitual do OE1 utilizando da técnica IDEF-Sim está representada na Figura 4.2.

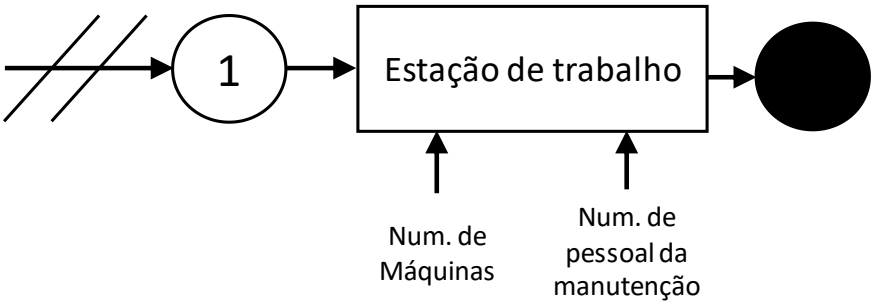

Figura 4.2 - Modelo conceitual para o OE1

4.1.2 Segundo objeto de estudo: Controle de estoque

O segundo objeto de estudo (OE2) usado para testar a estrutura de otimização é uma variedade teórica para o tamanho do lote econômico apresentada por Hillier e Lieberman (2015) e por Sifaleras e Konstantaras (2017). As encomendas chegam eletronicamente e aguardam para serem atendidas, respeitando um sistema FIFO (*first in, first out* - primeiro a entrar, primeiro a sair) para que um dos dois funcionários, quando disponível, atenda a demanda.

Cada pedido solicita uma quantidade de produtos e, se estiver disponível, o estoque é atualizado e o pedido é finalizado. Se não houver produto suficiente em estoque, o pedido aguardará até o final do processo de reabastecimento. Um terceiro empregado é responsável por verificar o estoque entre períodos definidos e gerar um reabastecimento de acordo com um tamanho de lote econômico definido, rodando a simulação pelo período de um ano. As variáveis têm suas distribuições estocásticas definidas na Tabela 4.2.

Tabela 4.2 - Distribuições estatísticas das variáveis do segundo objeto de estudo

Atividade	Distribuições de probabilidade (segundos)
Chegada da ordem	432+Normal (média=36, desvio padrão=10) – Exponencial (lambda=2)
Demandas por ordem	6+Normal (média=4, desvio padrão=1)
Tempo do serviço	(Demanda por ordem) / (3+Normal (média=2, desvio padrão=1))
Tempo de ressuprimento	271+Exponencial (lambda=10) – Exponencial (lambda=2)) × lot_{rep}

A resposta do problema consiste em determinar o ponto ótimo ($f(x^*)$) que é constituído pelas variáveis de decisão: tamanho do lote econômico, o nível de segurança que aciona um novo pedido e o período entre revisões para verificar o nível do lote. Todos esses parâmetros fazem parte da função objetivo (FO) e restrições apresentadas nas Equações 7 a 10, para maximizar a receita líquida. O parâmetro "$tempo_{espera}$" é dado pela soma do tempo total de espera que a solução atual gera para a ordem de compra entre a chegada e a saída da simulação.

$$\begin{aligned}\text{maximizar } FO(lot_i, sec_j, rev_k) \\ = \Bigg[& -1149557 - 1621 \times lot_i - 154 \times \sum tempo_{espera} + 1{,}8 \times seg_j \\ & - 3{,}9 \times {rev_k}^2 + 83{,}4 \times lot_i \times \sum tempo_{espera} \\ & - \left[\left((0{,}22 \times lot_i)^{\left(\frac{3}{\sum tempo_{espera}}\right)} \right) + 1 \right] - \left(0{,}0109 \times seg_j\right)^2 \\ & - 221000000 \times (lot_i)^2 \Bigg] / 45683000\end{aligned} \tag{7}$$

Sujeito a:

$$lot_i = \{100;\ 101;\ \ldots; 5000\}: \forall i \in lot\ \in \mathbb{Z} \tag{8}$$

$$seg_j = \{5;\ 6;\ \ldots; 95\}: \forall j \in seg\ \in \mathbb{Z} \tag{9}$$

$$rev_k = \{1800;\ 2100; 2400;\ \ldots; 28800\}: \forall k \in rev\ \in \mathbb{Z} \tag{10}$$

A Equação (7) resume todas as despesas e receitas relacionadas às vendas, aquisição, manutenção e geração de um pedido de reabastecimento e estoque. De acordo com as Equações (8-10), todas as variáveis são inteiras, com lot_i e seg_j relacionadas ao tamanho do lote econômico (medido em unidades) e nível de segurança (medido em porcentagem) respectivamente, sendo avaliadas com a adição de um elemento e rev_k o tempo entre as verificações no nível de estoque, com uma diferença de 300 segundos entre as soluções. Na Figura 4.3 é apresentada a representação gráfica do OE2.

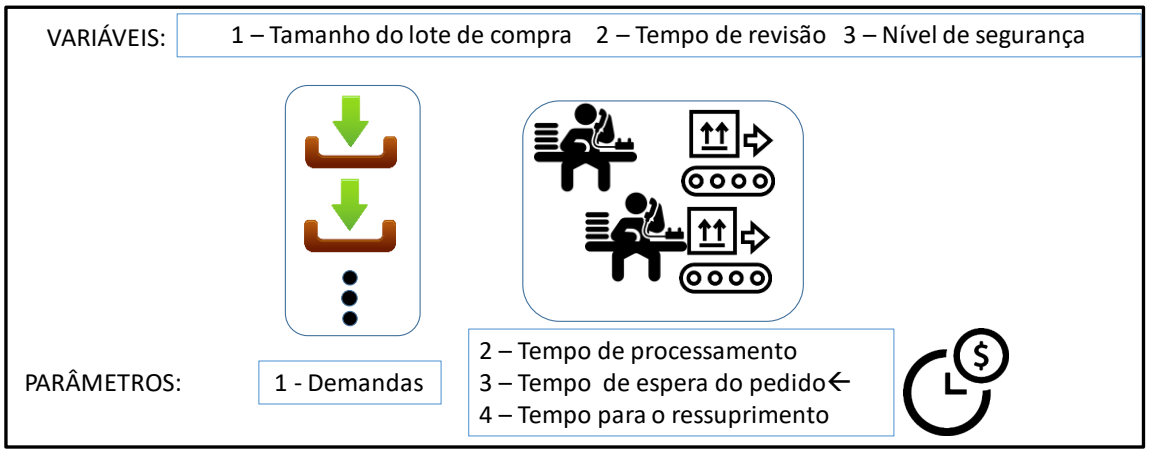

Figura 4.3 - Representação gráfica do OE2

Com as variáveis e restrições apresentadas, o problema tem um espaço de solução de 4901 x 91 x 91 = 40585181 cenários possíveis. Assim como no primeiro objeto de estudo, para se gerar todo o espaço solução, estima-se que seriam necessários no mínimo três anos de processamento, utilizando o hardware empregado no presente trabalho, fato este que também justifica a utilização de técnicas de otimização para busca no espaço solução. A modelagem conceitual do OE2 utilizando da técnica IDEF-Sim está representada na Figura 4.4.

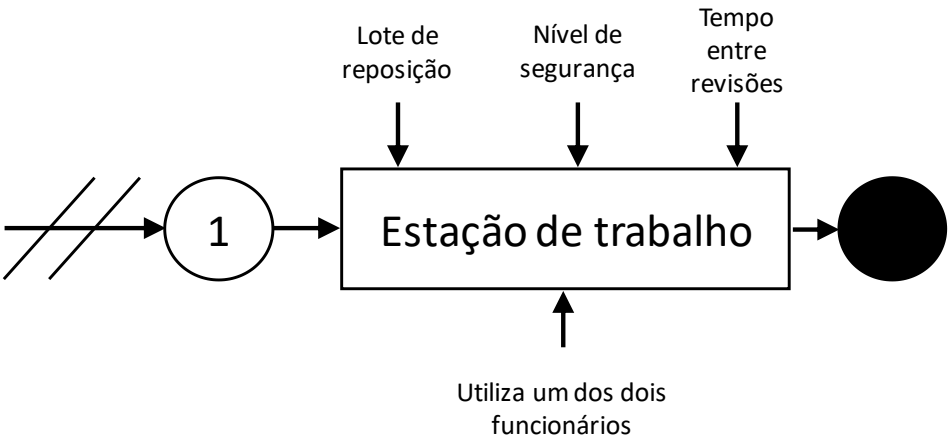

Figura 4.4 - Modelo conceitual para o OE2

4.1.3 Terceiro objeto de estudo: Máquinas e *buffers*

O terceiro objeto de estudo (OE3) usado para testar a estrutura de otimização foi retirado do livro Law e Kelton (2007). Este objeto de estudo considera uma linha de produção constituída por quatro estações de trabalho e três *buffers* (filas) de capacidade finita, com um

suprimento infinito de matéria prima sendo empurrada para a estação 1, assim como apresentado na Figura 4.5.

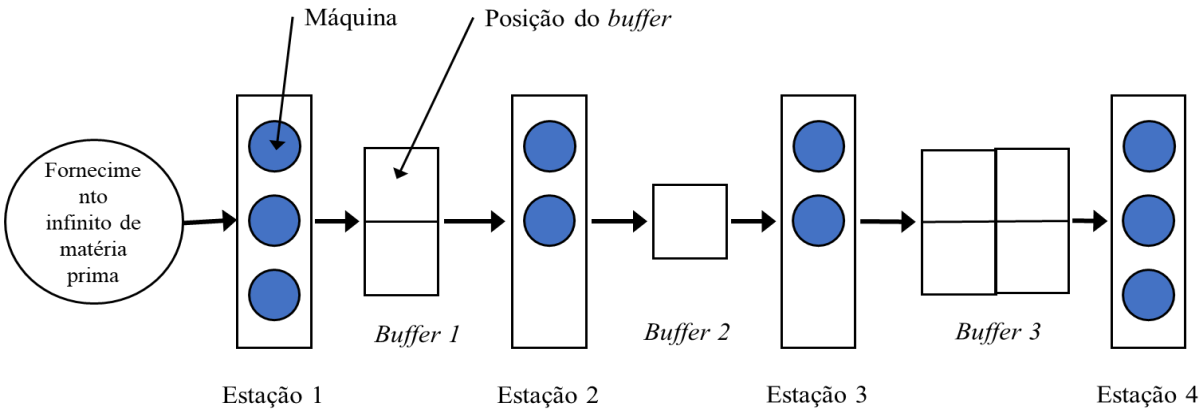

Figura 4.5 - Representação gráfica do OE3
Fonte: Adaptado de Law e Kelton (2007)

De acordo com a Figura 4.5, o produto é processado nas estações de 1 a 4 em sequência (com cada estação adicionando valor) e depois sai do sistema. Os tempos de processamento para uma máquina específica para cada estação estão especificados seguindo distribuições exponenciais que estão presentes na Tabela 4.3.

Tabela 4.3 - Distribuições estatísticas das variáveis do terceiro objeto de estudo

Atividade	**Distribuições de probabilidade (minutos)**
Máquina na Estação 1	Exponencial (20)
Máquina na Estação 2	Exponencial (30)
Máquina na Estação 3	Exponencial (12)
Máquina na Estação 4	Exponencial (15)

Fonte: Adaptado de Law e Kelton (2007)

Quando uma máquina nas estações 1, 2 e 3 acaba de realizar uma tarefa, empurra a produção para o *buffer* seguinte a menos que o mesmo esteja cheio. Nesse caso a máquina se torna "bloqueada" e não pode processar nenhum produto até que uma posição no *buffer* seguinte seja liberada. As máquinas da estação 1 só podem estar nos estados "ocupado" ou "bloqueado", as máquinas das estações 2 e 3 podem estar nos estados "ocupado", "bloqueado" ou "em espera", enquanto as máquinas da estação 4 usam dos estados "ocupado" ou "em espera".

O OE3 tem sete variáveis de decisão para serem definidas, sendo M_i (para i = 1, 2, 3 4) o número de máquinas para a estação i, e B_i (para i = 1, 2, 3) seja a capacidade de armazenamento do *buffer* i, sendo a capacidade das estações entre 1 e 3 máquinas e dos *buffers* entre 1 e 10 espaços. A resposta do problema consiste em determinar os valores de M_i e B_i que maximiza o lucro previsto para um período de 30 dias. Para o calculo da função objetivo, supõe-

se que o lucro de cada produto finalizado é de \$200, o custo de uma máquina ou de um espaço em cada *buffer* por 30 dias seja, respectivamente, de \$25000,00 e \$1000,00, sendo N o total de peças produzidas pela solução corrente assim como expresso nas Equações 11 a 13.

$$\text{maximizar } FO(M_i, B_i) = 200N - 25000\sum_{i=1}^{4} M_i - 1000\sum_{i=1}^{3} B_i \quad (11)$$

Sujeito a:

$M_i \in \{1;\ 2;\ 3\}$ para $i = 1, 2, 3, 4$ (12)

$B_i \in \{1;\ 2;\ \ldots; 10\}$ para $i = 1, 2, 3$ (13)

Como se está interessado no estado de regime permanente do sistema, foi simulado um período de 40 dias, com 10 dias de aquecimento (*warmup*) e 10 replicações para cada cenário. Utilizando o otimizador WITNESS®, os autores Law e Kelton (2007) encontraram uma resposta considerada ótima com as variáveis $M_1 = 3$, $M_2 = 3$, $M_3 = 2$, $M_4 = 2$, $B_1 = 7$, $B_2 = 9$, $B_3 = 4$, com um lucro esperado de \$587290,00. Para o OE3, são possíveis 3^4 x 10^3 = 81000 cenários em seu espaço solução. A modelagem conceitual do OE3 utilizando da técnica IDEF-Sim na Figura 4.6.

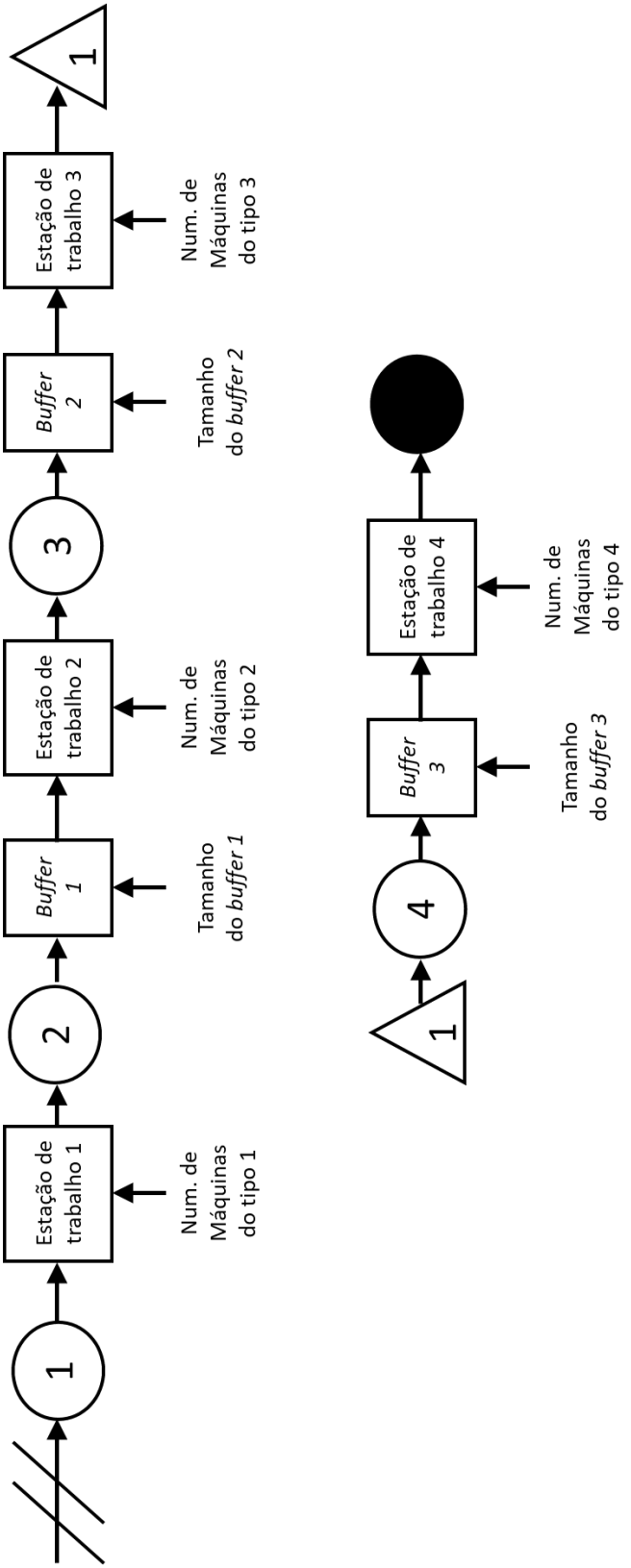

Figura 4.6 - Modelo conceitual para o OE3

4.2 Ambiente de otimização

Para o propósito do presente estudo, as características ilustradas na Figura 2.1 que apresentou os diferentes níveis iniciais a considerar em projetos de OvS, foram analisadas na seleção dos componentes necessários para estruturar o ambiente de otimização.

Considerando a definição dos objetos de estudo e os objetivos da tese, uma das solicitações para a operacionalização da presente tese é desenvolver e testar um ambiente camputacional capaz de executar os problemas de forma recursiva, recuperar dados relevantes para análises e retroalimentar o processo para orientar a busca da otimização através de áreas do espaço solução, a fim de encontrar a solução ótima ou próximo a ela.

Para isso, diferentes estratégias para pesquisar no espaço solução (metaheurística e aprendizagem de máquina) foram selecionadas. Para tal, apenas as melhores combinações de respostas mais prováveis de serem as melhores (previstas com técnicas de aprendizagem de máquina) foram executadas em paralelo (com computação paralela em sistemas *multi-core*), usando um ambiente capaz de lidar com todas essas questões em um mesmo software/hardware.

4.2.1 Seleção e teste das técnicas de aprendizagem de máquina

Durante as buscas e resultados gerados pela RSL, apresentados na Seção 2, não foram encontrados trabalhos em relação a junção de técnicas de metaheurística associada com aprendizado de máquina, aplicados a problemas de EP, apenas a comparação e uma metodologia em relação a outra. Como consequência, nenhum estudo foi encontrado como base para se saber quais seriam as técnicas de aprendizado de máquina mais adequadas para este caso.

Como referência foi utilizado o trabalho de Buitinck et al. (2013) que trata sobre a aplicação dos AMs de forma genérica, independente do problema. De forma resumida a forma como se deve usar os métodos de AM proposto pelos autores supracitados, está presente no fluxograma da Figura 4.7.

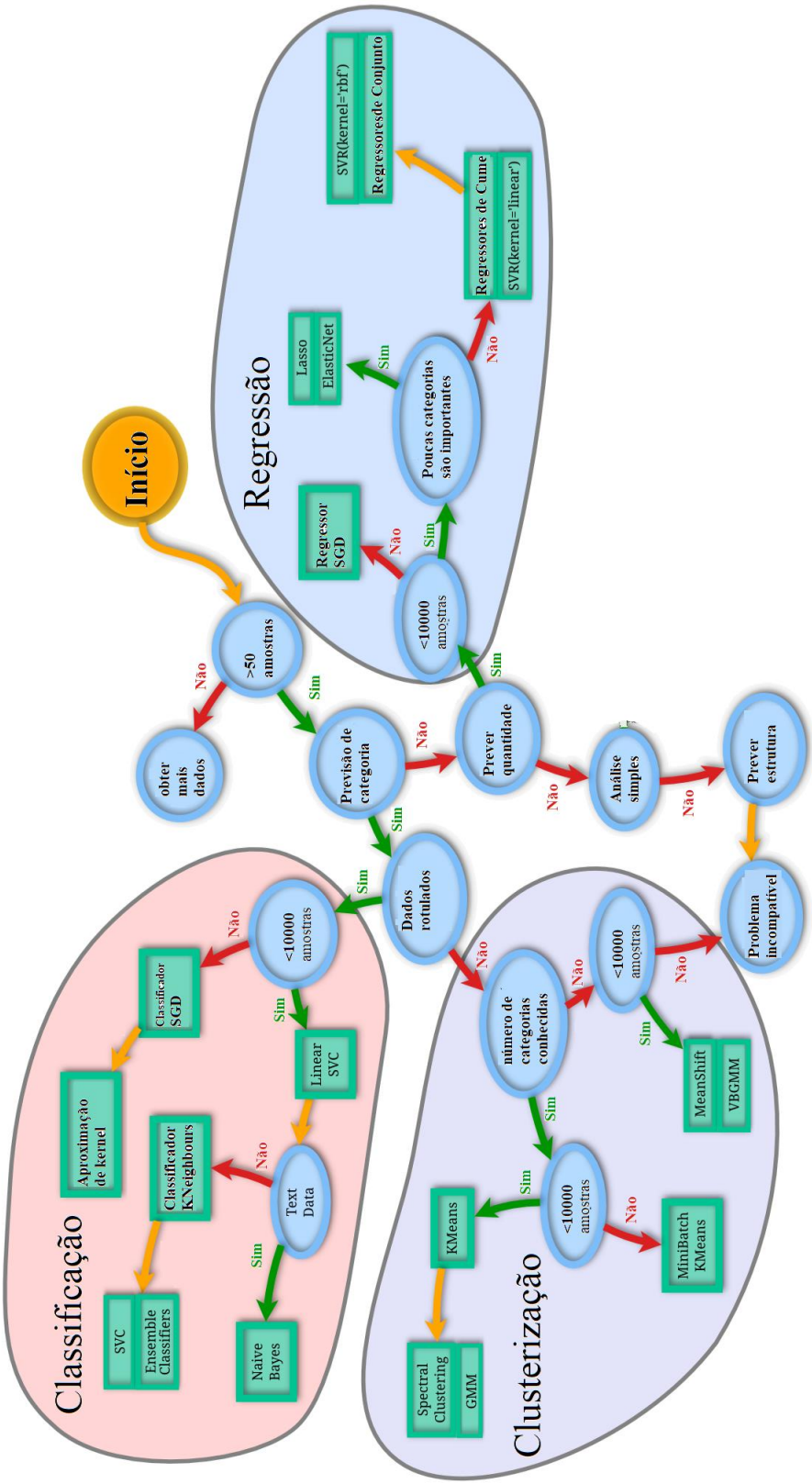

Figura 4.7 - Fluxograma de orientação para escolha de métodos de AM
Fonte: Adaptado de Sci-kit learning (2018)

Seguindo os passos do fluxograma presente na Figura 4.7, para os objetos de estudo, o primeiro passo é atendido pois suas respectivas bases de dados possuem mais de 50 amostras. No segundo passo é avaliado se o que se deseja gerar com a AM é uma previsão dentro de um número de categorias pré-determinadas ("dados rotulados") ou se deseja gerar a previsão em forma de um número ou quantidade ("Prever quantidade"). Para ambos os casos, as bases de dados podem passar pelo processo de redução da variabilidade/classificação de forma que os métodos de ambas as categorias podem ser aplicados no problema proposto (KUBAT, 2017).

Para avaliar as melhores técnicas de aprendizado de máquina, selecionou-se um total de 33 métodos dentre classificação e regressão (representando um conjunto expressivo das técnicas encontradas na literatura), pois todos os dados trabalhados eram rotulados, e de acordo com a Figura 4.7, não se justifica a atualização de técnicas de clusterização. Nos trabalhos de Buitinck et al., (2013) e Muller e Guido, (2017) foram relatados como esses métodos devem ser utilizados para se gerar os resultados pretendidos. A Tabela 4.4 mostra as famílias e seus respectivos métodos de AM considerados, bem como os níveis dos parâmetros utilizados e testados.

Tabela 4.4 - Métodos de AM e parâmetros testados no OE1, OE2 e OE3

Família	**Método**	**Parâmetros de ajuste utilizados**
Generalized Linear Models	*Ordinary Least Squares (OLS)*	----------
	Ridge Regression (RR)	*alpha = 0.5*
	Ridge Regression with Cross Validation (RRCV)	*alphas=[0.1, 1.0, 10.0]*
	LASSO	*alpha = 0.1*
	Elastic Net (EN)	*alpha=0.1, l1_ratio=0.7*
	Orthogonal Matching Pursuit (OMP)	*n_nonzero_coefs=3*
	Bayesian Ridge Regression (BRR)	----------
	Automatic Relevance Determination Regression (ARDR)	*compute_score=True*
	Logistic Regression (LR)	*C=100, penalty='l1', tol=0.01*
	Stochastic Gradient Descent (SGD)	*loss="hinge", penalty="l1",tol=0.01*
	Polynomial Regression (PR)	*('poly', PolynomialFeatures(degree=3)), ('linear', LinearRegression (fit_intercept=False))*
Linear and Quadratic Discriminant Analysis	*Linear Discriminant Analysis (LDA)*	*solver="svd", store_covariance=True*
Kernel ridge regression	*Kernel Ridge Regression (KRR)*	*alpha=1.0*
Support Vector Machines	*Support Vector Machines (SVM)*	----------

Família	**Método**	**Parâmetros de ajuste utilizados**
Nearest Neighbors	*Nearest Neighbors Classification (NNC)*	*n_neighbors=15, weights='uniform'*
	Nearest Centroid Classifier (NCC)	----------
Gaussian Processes	*Gaussian Process Regressor (GPR)*	*kernel = 1.0 * RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e3)) \ + WhiteKernel(noise_level=1e-5, noise_level_bounds=(1e-10, 1e+1)), alpha=0.0*
Naive Bayes	*Gaussian Naive Bayes (GNB)*	----------
	Multinomial Naive Bayes (MNB)	----------
	Bernoulli Naive Bayes (BNB)	----------
Decision Trees	*Decision Trees Classifier (DTC)*	*max_depth=None, min_samples_split=2, random_state=0*
	Decision Trees Regressor (DTR)	----------
Ensemble methods	*Bagging Classifier (BC)*	*KNeighborsClassifier(), max_samples=0.5, max_features=0.5*
	Random Forest Classifier (RFC)	*n_estimators=10*
	Random Forest Regressor (RFR)	*max_depth=2, random_state=0*
	Extra Trees Classifier (ETC)	*n_estimators=10, max_depth=None, min_samples_split=2, random_state=0*
	Ada Boost Classifier (ABC)	*n_estimators=100*
	Gradient Tree Boosting Regressor (GTBR)	*n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0*
	One-Vs-Rest Classifier (OVRC)	*LinearSVC(random_state=0)*
Multiclass and multilabel algorithms	*One-Vs-One Classifier (OVOC)*	*LinearSVC(random_state=0)*
	Error-Correcting Output-Codes (ECOC)	*LinearSVC(random_state=0), code_size=2, random_state=0*
Neural network models	*Multi-layer Perceptron Classifier (MLPC)*	*solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15,), random_state=1*
	Multi-layer Perceptron Regressor (MLPR)	*solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15,), random_state=1*

Fonte: Adaptado de Sci-kit learning (2018).

De todos os métodos apresentados na Tabela 4.4 que exigem um conjunto de parâmetros a serem especificados, apenas os padrões foram considerados e utilizados, de acordo com o pacote *SciKit-learn* do Python. Para mais detalhes sobre o significado dos parâmetros de cada método, a página do desenvolvedor que mantém o complemento Python, pode ser acessada (http://scikit-learn.org/stable/index.html). O uso desses valores padrões é devido ao objetivo principal do presente estudo em identificar os métodos mais adequados a serem usados e para dar uma boa visão sobre a direção da busca no espaço da solução, e não para ajustar um método de AM específico aos melhores parâmetros que irão ajustá-lo para uma melhor previsão, ficando a otimização dos parâmetros reservada para ser desenvolvida em trabalhos futuros.

Outros métodos foram identificados, mas não testados no presente artigo, porque exigem um tipo diferente de apresentação e manipulação de dados que diferem dos dados disponíveis sobre os objetos de estudo. Caso esses métodos fossem aplicados aos dados, gerariam um erro ou valores sem significado para a proposição do estudo. Exemplos de métodos de AM identificados e que não puderam ser testados são: *Least Angle Regression, Cross Decomposition, Gradient Tree Boosting Classifier, LARS LASSO, Cross Decomposition, Multi-output Regression* e *Multi-output Classifier* (BUITINCK et al., 2013).

As etapas a seguir foram aplicadas para gerar, tratar e fazer análises de dados para o problema proposto, tendo por base o trabalho de Kubat (2017), a fim de determinar a técnica de AM mais adequada a ser usada:

1) Definição da base de dados para treinamento e teste da AM: Os espaços solução dos problemas são definidos por 96800, 40585181 e 81000 possíveis soluções, respectivamente. Dois vetores foram construídos para ajudar na definição do banco de dados. O primeiro armazenou o espaço sequencial da solução e o segundo recebeu, de forma aleatória, a solução do primeiro vetor. Com este procedimento, o segundo vetor é aleatório e não possui valores repetidos. Para determinar uma amostra representativa, as primeiras 2000 soluções do segundo vetor (para os três objetos de estudo) foram selecionadas para representar a variabilidade do problema e podem ser geradas em um tempo viável no contexto apresentado.
2) Gerar dados e testar a compatibilidade com as técnicas de AM: Para aplicar um método de AM, é necessário ter informações especificadas em classes ou com certa repetição para que o cálculo das frequências relativas gere uma boa aderência ao banco de dados. Para o problema proposto, como em geral para problemas de SED, os resultados são discretos ou contínuos e não distribuídos uniformemente, o que originalmente gera muitas classes, e torna a utilização da AM um fator proibitivo, porque pode gerar treinamento e previsões ruins, não representando os dados com um fator mínimo de confiabilidade. Para averiguar essa questão, testes foram realizados para avaliar a possibilidade de utilização das técnicas de AM.
3) Treino e teste dos métodos e seleção do mais apto: Com a base de dados composta pelas 2000 soluções, 80% (1600) foram definidos para o treinamento e 20% (400) foram separados para a avaliação das previsões em relação aos valores reais. Para comparar os 33 métodos, foram utilizados os seguintes

critérios: intervalo de confiança (nível de significância γ = 95%) para a diferença entre o valor real e o previsto; k-fold com três níveis para verificar a presença de valores discrepantes; o menor valor máximo de erro (Min); o maior valor máximo para o erro (Max) gerados para o erro entre o predito e o real; o erro médio absoluto (*mean absolute error*, MAE) como a média do somatório dos valores absolutos da diferença entre o real e o previsto, é o valor médio esperado para o erro na resposta da previsão, portanto quanto mais perto do valor "0", melhor é o índice; o erro médio quadrático (*mean squared error*, MSE), assim como o MAE, um valor de MSE próximo de "0" é desejável; e o tempo real (*wall clock*) necessário para o treinamento e a previsão. Para classificar os melhores resultados, os valores de MAE e MSE foram considerados, uma vez que o método mais desejável deve ter o erro total mínimo de predição.

A seleção dos métodos MAE e MSE se deve pela necessidade de se avaliar o erro médio gerado pela aplicação da AM e escolha do mesmo que gere menor erro possível (POSPIESZNY; CZARNACKA-CHROBOT; KOBYLINSKI, 2018). As Equações 14 e 15 apresentam o cálculo dos mesmos.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \tag{14}$$

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2 \tag{15}$$

Sendo:
n é a quantidade de soluções a serem prevista/classificadas;
y_j é a j-ésima solução;
$\hat{y}_j$ é a j-ésima previsão para a y_j.

A diferença básica entre as métricas MAE e MSE é que a MSE dá uma maior ênfase para números discrepantes (*outliers*), assim é possível avaliar a presença dos mesmos. Para seleção dos melhores métodos de AM, foi adotado como critério de ordenação o MAE como métrica principal, de maneira tal que o seu menor valor determina o grau de assertividade do método.

4.2.2 Seleção e teste das metaheurísticas e paralelismo

Na literatura científica, muitos métodos de otimização via simulação a eventos discretos foram utilizados com sucesso. Exemplos disso incluem heurísticas, metaheurísticas, meta-modelos (modelos substitutos), classificação e seleção, enumeração completa e caixa-preta

(software comercial) e outros, como mostrado na Figura 2.4 (FU, 1994a; GRULER et al., 2018; JUAN et al., 2015; KLEIJNEN, 2017; LONG-FEI; LE-YUAN, 2013; NEGAHBAN; SMITH, 2014; RILEY, 2013; XU et al., 2015). A seleção de um método específico está relacionada ao tamanho do espaço da solução e ao conhecimento prévio de como as variáveis interagem entre si. Neste caso, pesquisar todo o espaço da solução não é feito com um tempo viável em uma rotina diária.

Considerando os métodos anteriormente citados, optou-se pela metodologia de otimização metaheurística pois, segundo os autores supracitados, esses métodos de otimização consideram critérios de parada que envolvem alguma restrição em seus elementos fundamentais, por exemplo, com o tempo total da simulação (*wall clock*) ou número máximo de interações sem melhoria. Outro elemento é a geração de um número de soluções por cada interação do algoritmo de busca que é orientado pela proposição do método. Essas duas características (número de soluções por iteração e avaliação de soluções distintas) atendem ao cumprimento dos objetivos desejáveis do presente estudo, de modo a caminhar no espaço solução de forma paralela.

A área de simulação paralela é uma assunto que remonta sua origem à década de 1970 (FUJIMOTO et al., 2017). Porém, consideram as mudanças computacionais, referente a computadores modernos (a partir do ano 2003) com a adoção de forma mais barata (financeiramente) de processadores *multi-core* em computadores pessoais (KIRK; HWU, 2010). Estes são capazes de realizar tarefas paralelas a um custo mais baixo em relação à equipamentos construídos e dedicados unicamente para tal finalidade. No entanto, nem todos os programas têm a capacidade de aproveitar este método, mas isso depende do conhecimento prévio e da possibilidade de distribuir um estudo com pouca ou nenhuma dependência de dados anteriores do programa que justifique o uso do paralelismo (CHOI; SEO; KIM, 2014).

No que se refere à otimização, é possível utilizar o paralelismo para avaliar múltiplos cenários ao mesmo tempo (SAVINIEC; SANTOS; COSTA, 2018). Para este propósito, o Algoritmo Genético (GA) e o Procedimento de Pesquisa Adaptativa Aleatorizada Gulosa (GRASP) foram escolhidos como metaheurísticas que usam de busca populacional para a fase de geração de solução e caminhada no espaço solução. Esses dois métodos de otimização foram usados antes da simulação, no entanto, nenhum trabalho foi encontrado que unisse em um mesmo ambiente paralelismo, metaheurística e aprendizado de máquina, mas sim em trabalhos distintos, a exemplo de JIA et al. (2015), RESENDE e RIBEIRO (2016), SONG, QIU e LIU (2015) e TIACCI (2015).

A comparação entre GA e AM não é uma tarefa inédita, a qual já foi identificado trabalho relacionando o ajuste de parâmetros em células de manufatura (VOSNIAKOS; TSIFAKIS; BENARDOS, 2005) e a previsão usando GA e *Artificial Neural Networks* (Redes Neurais Artificiais - ANN) com GA foram mais lentas, mas com melhor confiança de resposta do que ANN (CAN; HEAVEY, 2012). Para exemplos de problemas de Engenharia de Produção usando GRASP é possível citar a distribuição de óleo petroquímico, controle de estoque e roteamento de veículos (MUJICA MOTA; FLORES DE LA MOTA, 2017), bem como a integração de GRASP e mineração de dados (RIBEIRO; PLASTINO; MARTINS, 2006). Esses dois métodos de otimização (GA e GRASP) foram selecionados porque ambos usam a pesquisa populacional para percorrer o espaço solução, tornando isso um ponto de partida para comparar diferentes métodos.

A fim de se gerar um ponto de comparação com maior ênfase no método de AM e paralelismo, as metaheurísticas GA e GRASP foram aplicadas no OE1 e o método com melhor resultado aplicado no OE2 e OE3, com um maior número de níveis de previsão, para se testar a influência desse ajuste para o resultado da OvS do OE2 e OE3.

As etapas a seguir foram aplicadas para teste e análise dos dados gerados a partir da implementação das metaheurísticas e paralelismo, juntamente com o método de AM selecionado:

1. Geração de um ponto de comparação: Para se avaliar a qualidade da resposta gerada pela otimização, duas referências foram utilizadas. Para o primeiro e terceiro objeto de estudo, foi possível gerar todo o espaço solução, encontrado o ótimo global para o problema, necessitando respectivamente de 9 a 20 dias para o seu término. Para o segundo objeto de estudo foram geradas e avaliadas uma grande quantidade de soluções no intervalo de tempo ininterrupto de 18 dias, encontrando assim um ótimo local de referência.
2. Seleção e avaliação de desempenho em diferentes hardwares para cenários de otimização: Os hardwares escolhidos para os experimentos foram dois computadores com diferentes configurações. O primeiro foi um Dell XPS 8910 com 16 GB de RAM equipado com processador Intel i7 7700 com 4 processadores e 8 *threads*, executando uma carga de processamento paralela com uma velocidade de 4.00 GHz para todos os núcleos e de 4.16 GHz em processamento de instrução sequencial em um núcleo. A segunda máquina foi um Supermicro X9DAi com dois Intel Xeon E5-2620, executando uma carga de processamento paralela com uma velocidade de 2,29 GHz para todos os núcleos

e 2,48 GHz em processamento sequencial em um único núcleo, com um total de 12 núcleos, 24 *threads* e 96 GB de RAM. Este conjunto de máquinas foi utilizado para avaliar o impacto do poder de computação nos resultados da otimização em termos de quantidade de núcleos *versus* a velocidade de processamento. Para garantir que os mesmos resultados sejam obtidos independentemente do software utilizado, ambos os sistemas utilizados foram Windows 10 x64 versão 1709 com todas as atualizações até 01 de maio de 2018, rodados com a mesma semente aleatória, tendo por parâmetro de comparação o tempo para as instâncias.

3. Geração e teste de cenários com as técnicas de otimização: Para as técnicas de otimização selecionadas, diferentes configurações podem ser geradas, com o intuito de testar a influência de um fator em detrimento de outro. A exemplo, não se sabe qual o efeito de se gerar uma maior ou menor quantidade de previsões via AM, para a qualidade de resposta final da otimização.

De forma geral, o método de otimização proposto visa avaliar a qualidade da resposta gerada no final da otimização, em termos de "distância" em relação à melhor solução de referência e o tempo necessário para se encontrar o mesmo.

4.3 Construção do ambiente de geração e coleta de dados para otimização

Para atingir os objetivos do presente estudo, duas abordagens diferentes para otimização via SED foram aplicadas relacionadas com o software: metaheurística e aprendizado de máquina, e hardware com o uso de paralelização em dois tipos diferentes de hardware. Assim, o ambiente proposto possui cinco características: 1) programar duas metaheurísticas, 2) programar 33 técnicas de aprendizado de máquina, 3) executar um ambiente de modelagem computacional de SED, 4) executar os cenários de SED escolhidos de forma serial e paralela e 5) unir no mesmo ambiente as quatro características citadas anteriormente para coleta e análise dos dados.

Considerando o requisito seleção de *software*, o *framework* usou pacotes de código aberto encontrados nos repositórios oficiais, que foram construídos usando a seguinte configuração:

- Ambiente computacional de programação em Python 3.6.5 x64, que possui a função *MapReduce* incorporada para processamento paralelo, sendo esta uma condição básica e essencial para o presente trabalho. Esta função específica tem

a capacidade de "Mapear" a quantidade de processamento em paralelo que o computador pode processar, distribuir as tarefas entre os diferentes "núcleos" de processamento e depois agrupar todas as informações geradas. Este ambiente de programação usa o gerenciamento de pacotes do Pip 9.0.3 para baixar e instalar os recursos necessários e ou definidos pelo usuário. Inicialmente, outros ambientes de programação foram considerados, mas nenhum apresentou as cinco características desejadas (DAGKAKIS; HEAVEY, 2016). Como vantagem, uma alternativa para uma linguagem compilada (ex. Java e C ++), o Python é uma linguagem interpretada que torna mais fácil para um programador iniciante desenvolver um código e/ou encontrar erros (RASCHKA; JULIAN; HEARTY, 2016), fato este que auxilia na programação das metaheurísticas utilizadas. Para o propósito deste estudo, a característica de "facilidade" de reprodução da estrutura apresentada é preferível a uma pequena melhoria computacional que outras linguagens possam oferecer, tornando o presente trabalho reproduzível para uma maior quantidade de pesquisadores ou práticos da área;

- Para a simulação a eventos discretos, foi selecionado o ambiente SimPy 3.0.10, que permite modificar os parâmetros do problema a partir do código gerado na otimização que já foi testada e utilizada em problemas de engenharia (HADJSAID et al., 2009; KOVALCHUK et al., 2018; PINHO et al., 2018; VÉJAR; CHARPENTIER, 2012);
- Para tornar possível a base de dados para aprendizagem de máquina com treinamento e previsão, os seguintes pacotes foram usados: NumPy 1.14.3, SciKit-Learn 0.19.1, Pandas 0.22.0 (com as dependências Python-dateutil-2.7.2, pytz-2018.4 e seis-1.11.0) e SciPy 1.0.1. Estes pacotes são usados em estudos para aprendizagem de máquina usando linguagem Python e já possuem AM disponível para testes (MÜLLER; GUIDO, 2017; PEDREGOSA et al., 2011; RASCHKA; JULIAN; HEARTY, 2016);
- De forma complementar para a coleta e tratamento de dados, arquivos CSV ("*comma separated values*") e TXT ("*eletronic text file*") foram gerados para armazenamento dos dados. A representação gráfica dos resultados foi realizada utilizando o pacote MatPlotLib 2.2.2 (e as dependências cycler-0.10.0, kiwisolver-1.0.1 e pyparsing-2.2.0).

A partir da definição do ambiente de programação a ser utilizado, algoritmos foram idealizados para compreender fatores necessários para se atender as cinco características desejadas para a otimização.

4.3.1 Algoritmos para gerar solução referência para os objetos de estudo

Na abordagem para programar a otimização, sete algoritmos foram construídos, dos quais os três primeiros foram usados para construir as populações iniciais e a coleta de dados primários para o banco de dados dos métodos de AM, com os respectivos treinamentos, previsões e análise dos dados de saída, e os quatro algoritmos seguintes relacionam a integração das metaheurísticas, AM e paralelismo com os objetos de estudo. Na Tabela 4.5, o Algoritmo 1 apresenta a modelagem computacional para a geração de todo o espaço solução para o primeiro objeto de estudo.

Tabela 4.5 - Algoritmo 1 - Geração do espaço solução para o OE1

Início

Variáveis de entrada:
$M_i = \{1;\ 2;\ \ldots;\ 40\}$: % Conjunto de máquinas
$P_j = \{1;\ 2;\ \ldots;\ 20\}$: % Conjunto de pessoal da manutenção
$T_k = \{30;\ 31;\ \ldots;\ 150\}$: % Conjunto de tempos de simulação
% Fase 1: Geração da simulação de eventos discretos
1 $simulAlocaçãoRecursos \leftarrow (M_i; P_j; T_k)$
2 $t \leftarrow 0$ % toda simulação começa no tempo 0
enquanto $t \leq T_k$ **faça**
3 $tempoProcessamento \leftarrow rand(Normal[10,2])$ para cada M_i
4 $tempoEntreQuebras \leftarrow rand(Expo[1/300])$ para cada M_i
5 $tempoReparo \leftarrow const(30)$ para casa M_i usando um P_j
6 $\sum(M_i\ produtos\ produzidos\ no\ tempo\ t)$
7 $t \leftarrow t + avançaTempoPróximoEvento$
Fim
8 gravaResultados $\leftarrow (M_i; P_j; T_k; \sum(produtos))$ %salva os dados em um bloco de notas
9 **retorna** FO ($\sum(produtos)$) % Retorna a FO de acordo com a Equação 3
% Fase 2: Recursividade para a geração do espaço solução do primeiro objeto de estudo
10 **para cada** M_i**faça**
11 **para cada** P_j**faça**
12 **para cada** T_k**faça**
13 $simulAlocaçãoRecursos \leftarrow (M_i; P_j; T_k)$
14 gravaResultados $\leftarrow (M_i; T_k;\ FO)$ % salva os dados em um bloco de notas
Fim
Fim
Fim
% Fase 3: Análise dos dados e estatística básica
15 % desenho do espaço solução
16 % encontrar a melhor combinação de $(M_i; P_j; T_k;\ FO)$

Analisando o Algoritmo 1, nas linhas 1 a 8 é gerado o ambiente de SED em forma de função (primeira fase) a ser chamada recursivamente pelo gerador de todas as combinações de possíveis soluções, presente nas linhas 10 a 14 (segunda fase). A partir dos dados armazenados (linha 14),) estatísticas básicas são aplicadas para avaliar qual a melhor resposta que representa o ótimo global do problema (terceira fase).

O Algoritmo 2 presente na Tabela 4.6, exibe de forma diferente do Algoritmo 1, a busca pela solução de comparação para o segundo objeto de estudo.

Tabela 4.6 - Algoritmo 2 - Geração de ótimo local para o OE2

Início

Variáveis de entrada:
$lot_i = \{100;\ 101;\ \ldots; 5000\}$: $\forall i \in lot$ %Conjunto de quantidade de itens de reposição
$seg_j = \{5;\ 6;\ \ldots, 95\}$: $\forall j \in seg$ %Conjunto de níveis de segurança
$rev_k = \{1800;\ 2100; 2400;\ \ldots; 28800\}$: $\forall k \in rev$ %Conjunto de tempos entre revisões
% Fase 1: Recursividade utilizando busca randômica para o segundo objeto de estudo
1 $\boldsymbol{simulLotEconomico} \leftarrow (semente; lot_i; seg_j; rev_k)$
2 $t \leftarrow 0$ % toda simulação começa no tempo 0
enquanto $t \leq X_{max}$ **faça**
3 $tempoProcess \leftarrow rand.distribution$ %para cada variável de acordo com a Tabela 4.2
4 $\sum (tempo_{espera}\ no\ tempo\ t)$
5 $t \leftarrow t + avançaTempoPróximoEvento$
Fim
6 **retorna** FO de acordo com a Equação (7)
% Fase 2: Recursividade para a geração do espaço solução do primeiro objeto de estudo
7 $parada == False$ %Definição de critério de parada
8 **enquanto** $parada == False$ **faça** %Critério de parada para a geração de soluções
9 **enquanto** $rep \leq rep_{max}$ **faça**
10 $lot_i = rand\ \{100;\ 101;\ \ldots; 5000\}$
11 $seg_j = rand\ \{5;\ 6;\ \ldots, 95\}$
12 $rev_k = rand\ \{1{,}800;\ 2100; 2400;\ \ldots; 28800\}$
13 $simulLotEconomico \leftarrow (semente; lot_i; seg_j; rev_k)$
14 gravaResultados $\leftarrow (lot_i, seg_j, rev_k, FO)$ % salva os dados em um bloco de notas
15 $rep = rep + 1$
Fim
16 **se (**$parada > X_{max}$**)**; $parada == True$ %Parada ativada pelo usuário
Fim
% Fase 3: Análise dos dados e estatística básica
17 % desenho do espaço solução
18 % encontrar a melhor combinação de $(lot_i; seg_j; rev_k;\ FO)$

O Algoritmo 2 apresenta a modelagem computacional para a geração de uma grande quantidade de soluções por uma busca randômica do segundo objeto de estudo. Na primeira fase (linhas 1 a 6), o ambiente de modelagem computacional de SED é parametrizado para gerar o segundo objeto de estudo. Na segunda fase (linhas 7 a 16), é gerado um ambiente de busca randômica para a geração de uma grande quantidade de soluções, de forma a caminhar no espaço solução, de forma a gerar uma amostra representativa do mesmo e obter uma solução

ótima local, tendo em vista que é impossível com os recursos disponíveis, a geração de todo o espaço solução.

De forma diferente que o Algoritmo 1, a segunda fase do Algoritmo 2 foi construída considerando dois fatores. O primeiro refere-se à inserção da possibilidade de se passar como parâmetro da simulação a semente geradora dos números aleatórios (linha 13). O segundo fator remete a necessidade de que em trabalhos de SED, é comum a utilização de um número de replicações para se alcançar determinado nível de variância para os dados gerados (linha 9). O critério de parada da busca de um ótimo local para referência (linha 16) foi definido a partir de estudos exploratórios, quando do resultado da sua implementação apresentada no Capítulo 5. Com os dados gerados, estatísticas básicas foram coletadas para se encontrar o ótimo local do segundo objeto de estudo. De maneira análoga ao Algoritmo 1, o Algoritmo 3 presente na Tabela 4.7 foi desenvolvido para se gerar todo espaço solução para o OE3.

Tabela 4.7 - Algoritmo 3 - Geração de ótimo global para o OE3

Início

Variáveis de entrada:
$M_i = \{1;\ 2;\ 3; 4\}$: % Conjunto de máquinas para a estação i.
$B_j = \{1;\ 2;\ \ldots;\ 10\}$: % Conjunto de espaços para o *buffer* j.
% Fase 1: Geração da simulação de eventos discretos
1 $simulMaquinasBuffer \leftarrow (M_i; B_j)$
2 $t \leftarrow 0$ % toda simulação começa no tempo 0
enquanto $t \leq T_k$ **faça**
3 $tempoProcessamentoM_1 \leftarrow rand(Expo[20])$.
4 $tempoProcessamentoM_2 \leftarrow rand(Expo[30])$
5 $tempoProcessamentoM_3 \leftarrow rand(Expo[12])$
6 $tempoProcessamentoM_4 \leftarrow rand(Expo[15])$
7 $\sum(M_i\ produtos\ produzidos\ no\ tempo\ t)$
8 $t \leftarrow t + avançaTempoPróximoEvento$
Fim
9 gravaResultados $\leftarrow (M_i; B_j; \sum(produtos))$ %salva os dados em um bloco de notas
10 **retorna** FO $(200 \sum(produtos) - custos)$ % Retorna a FO de acordo com a Equação 11
% Fase 2: Recursividade para a geração do espaço solução do primeiro objeto de estudo
11 **para cada** M_1**faça**
12 **para cada** B_1**faça**
13 **para cada** M_2**faça**
14 **para cada** B_2**faça**
15 **para cada** M_3**faça**
16 **para cada** B_3**faça**
17 **para cada** M_4**faça**
18 $simulMaquinasBuffer \leftarrow (M_i; B_j; T_k)$
19 gravaResultados $\leftarrow (M_i; B_j;\ FO)$ % salva os dados em um bloco de notas
Fim
Fim
Fim
Fim
Fim
Fim
Fim
Fim
% Fase 3: Análise dos dados e estatística básica
20 % desenho do espaço solução
21 % encontrar a melhor combinação de $(M_i; B_j;\ FO)$

De forma análoga ao Algoritmo 1, no Algoritmo 3, as linhas de 1 a 8 geram a modelagem computacional para o OE3. Das linhas 11 a 19, todo o espaço solução é gerado para o problema, interagindo de forma recursiva para se obter todas as combinações possíveis para o OE3 denominado pela função "$simulMaquinasBuffer$".

4.3.2 Algoritmos para otimização dos objetos de estudo

Para programar a otimização da SED, três algoritmos foram construídos, dos quais o primeiro foi usado para construir as populações iniciais e a coleta de dados primários para o banco de dados da AM, para respectivamente treino e previsão, apresentado na Tabela 4.8.

Tabela 4.8 - Algoritmo 4 - Geração da população inicial e banco de dados para o OE1

Início

Variáveis de entrada:
$M_i = \{1;\ 2;\ \ldots;\ 40\}$: % Conjunto de máquinas
$P_j = \{1;\ 2;\ \ldots;\ 20\}$: % Conjunto de pessoal da manutenção
$T_k = \{30;\ 31;\ \ldots;\ 150\}$: % Conjunto de tempos de simulação
% Fase 1: Geração da simulação de eventos discretos
1 $simulAlocaçãoRecursos \leftarrow (M_i; P_j; T_k)$
2 $t \leftarrow 0$ % toda simulação começa no tempo 0
enquanto $t \leq T_k$ **faça**
3 $tempoProcessamento \leftarrow rand(Normal[10,2])$ para cada M_i
4 $tempoEntreQuebras \leftarrow rand(Expo[1/300])$ para cada M_i
5 $tempoReparo \leftarrow const(30)$ para cada M_i usando um P_j
6 $\sum(M_i\ produtos\ produzidos\ no\ tempo\ t)$
7 $t \leftarrow t + avançaTempoPróximoEvento$
Fim
8 gravaResultados $\leftarrow (M_i; P_j; T_k; \sum(produtos))$ %salva os dados em um bloco de notas
9 **retorna** FO ($\sum(produtos)$) % Retorna a FO de acordo com a Equação 3
% Fase 2: Construção da população inicial
8 **Para cada** $vetPop$ **faça**
9 $vetPop[12{,}5\%] \leftarrow rand(A\_M_i) + rand(A\_P_j) + rand(A\_T_k)$
10 $vetPop[12{,}5\%] \leftarrow rand(B\_M_i) + rand(A\_P_j) + rand(A\_T_k)$
11 $vetPop[12{,}5\%] \leftarrow rand(A\_M_i) + rand(B\_P_j) + rand(A\_T_k)$
12 $vetPop[12{,}5\%] \leftarrow rand(A\_M_i) + rand(A\_P_j) + rand(B\_T_k)$
13 $vetPop[12{,}5\%] \leftarrow rand(B\_M_i) + rand(B\_P_j) + rand(A\_T_k)$
14 $vetPop[12{,}5\%] \leftarrow rand(B\_M_i) + rand(A\_P_j) + rand(B\_T_k)$
15 $vetPop[12{,}5\%] \leftarrow rand(A\_M_i) + rand(B\_P_j) + rand(B\_T_k)$
16 $vetPop[12{,}5\%] \leftarrow rand(B\_M_i) + rand(B\_P_j) + rand(B\_T_k)$
Fim
% Fase 3: Função de avaliação para a população inicial
17 **para cada** $vetPop$ **faça**
18 $resultPop \leftarrow simulAlocaçãoRecursos(vetPop)$
Fim
% Fase 4: Tratamento dos dados para treinamento inicial da AM
19 **para cada** $vetPop$ **faça**
20 $dadosAM \leftarrow resultPop\ MOD 300$ % Transforma os dados em classes
21 $marcaçõesAM \leftarrow vetPop$
Fim
% Fase 5: Ordena a população e defina os critérios de parada
22 $vetPop \leftarrow ordenarMaxParaMin(vetPop;\ resultPop)$
23 $iterMax \leftarrow 11$
24 $iter \leftarrow 0$

No Algoritmo 4 da Tabela 4.8, a primeira fase é utilizada para construir um modelo de simulação a eventos discretos (linhas 1 a 7) que possa ser recursivamente chamado para construir a população inicial e avaliar os cenários selecionados para a próxima geração, neste caso para o OE1. Em particular, o modelo de simulação foi desenvolvido para permitir uma execução em sequência ou de modo paralelo. Na segunda fase (linhas 8 a 16), uma população inicial foi definida para gerar 120 soluções aleatórias, respeitando uma combinação de níveis

baixos e altos para as variáveis do problema. Neste caso, a variável "máquinas" tem um intervalo de valores de 1 a 40, o baixo ($B\_$) representa valores de 1 a 19 e o alto ($A\_$) de 20 a 40 com a mesma analogia para as variáveis pessoal e tempo. Esse arranjo foi definido em vez de usar uma geração aleatória para a população inicial, a fim de dar ao método AM um quadro de dados inicial que possibilite uma boa representatividade do espaço da solução com uma distância mais uniforme e esparsa entre as soluções geradas. Como consequência, eles têm uma chance maior de gerar previsões próximas aos valores reais com baixos erros para o treinamento da AM.

A terceira fase é a avaliação das soluções (linhas 17 e 18). Isso pode ser feito de maneira serial ou paralela. A chamada de simulação é a única parte que é deliberadamente executada desta maneira, porque é a função do programa que mais demanda poder computacional, como consequência do tempo para ser processado e é uma função recursiva chamada cada vez que o valor final de uma solução deve ser conhecido. A quarta fase é a construção e treinamento para o método de AM (linhas 19 a 21). Para reduzir a variação da função de avaliação retornada pela simulação antes de entrar no quadro de dados, é possível a aplicação da transformação dos dados, na qual o resultado da simulação pode ser substituído, este fator é melhor explorado no Capítulo 5, com os resultados da otimização.

A última fase do Algoritmo 4 é ordenar o vetor solução para a população inicial, de acordo com os valores correspondentes armazenados no vetor de resultados, de maneira decrescente para o valor dos resultados. Essa ordenação é necessária porque ambos os algoritmos de otimização aproveitam no seu processo de otimização o vetor ordenado das respostas (linha 22). As duas últimas funções (linhas 23 e 24) definem os parâmetros para os critérios de parada do processo de otimização. Em nossos programas, a pesquisa de otimização foi definida para interromper o programa quando 11 interações consecutivas são geradas sem melhoria na melhor solução. Com os critérios de parada definidos, a avaliação de desempenho dos algoritmos está relacionada à comparação de tempo para gerar a solução e qualidade da solução referente a sua distância para o ótimo global (primeiro objeto de estudo).

O Algoritmo 4 foi utilizado também para a geração da população inicial e banco de dados para os objetos de estudo 2 e 3, com as devidas mudanças nas linhas 1 a 9. Foi utilizado o código das linhas 1 a 5 do Algoritmo 2 para se chamar recursivamente a simulação do objeto de estudo 2 pela função "$simulLotEconomico$", assim como as linhas 1 a 8 do Algoritmo 3 para o OE3, chamando a função "$simulMaquinasBuffer$".

Depois da geração da base de dados inicial pela população inicial e com o treinamento do método de AM, o próximo passo é o teste do algoritmo de otimização. O Algoritmo 5 presente na Tabela 4.9 mostra a otimização por GA, com e sem AM.

Tabela 4.9 - Algoritmo 5 - Algoritmo Genético com e sem AM

Início

enquanto $iter \leq iterMax$ **faça**
  para $numOffspr$ **faça**
    % Fase 1: Seleção e crossover
    $a \leftarrow rand(vetPop)$
    $b \leftarrow rand(vetPop)$
    $crossPont \leftarrow rand(3)$
    se $crossPont = 1$ **faça**
      $offspr1 \leftarrow M_b + S_a + T_a$
      $offspr2 \leftarrow M_a + S_b + T_b$
    Fim
    se $crossPont = 2$ **faça**
      $offspr1 \leftarrow M_a + S_b + T_a$
      $offspr2 \leftarrow M_b + S_a + T_b$
    Fim
    se não
      $offspr1 \leftarrow M_a + S_a + T_b$
      $offspr2 \leftarrow M_b + S_b + T_a$
    Fim
    % Fase 2 Mutação
    $c \leftarrow rand(mutProb)$
    se $mutProb \geq c$ **faça**
      $offspr1 \leftarrow mutCrossov(rand(restrito))$
    Fim
    $c \leftarrow rand(mutProb)$
    se $mutProb \geq c$ **faça**
      $offspr2 \leftarrow mutCrossov(rand(restrito))$
    Fim
  Fim
  % Fase 3.1: Avaliação e próxima geração sem AM
  para cada $offspr$ **faça**
    $resultPop(rand) \leftarrow simulAlocaçãoRecursos(offspr)$
    $vetPop(rand) \leftarrow offspr$
  Fim
  % Fase 3.2: Avaliação e próxima geração com AM
  para cada $offspr$ **faça**
    $tempResult \leftarrow prevAM(offspr)$
    $tempVet \leftarrow offspr$
  Fim
  $tempResult \leftarrow ordenMaxParaMin(tempVet, tempResult)$
  para cada $tempResult(melhor)$ **faça**
    $resultPop(rand) \leftarrow simulAlocaçãoRecursos(tempResult)$
    $vetPop(rand) \leftarrow tempVet$
  Fim
  % Fase 3.2.1: Tratamento dos dados para melhorar a AM
  para cada $tempResult(melhor)$ **faça**
    $dataML \leftarrow dataML + tempResult(melhor)\ MOD300$
    $markML \leftarrow markML + tempVet$
  Fim
  % Fase 4: Melhor solução corrente e contador de interações

```
35      starResultAntes ← resultPop(primeiro)
36      vetPop ← ordenMaxParaMin(vetPop, resultPop)
37      starResultDepois ← resultPop(primeiro)
38      se starResultDepois ≥ starResultAntes faça
39          iter ← 0
        Fim
40      se não
41          iter ← iter + 1
        Fim
    Fim
```

A fase 1 do Algoritmo 5 (linhas 4 a 15 da Tabela 4.9) representa a seleção e o cruzamento (operação de *crossover*) no Algoritmo Genético. A seleção em si é um sorteio aleatório de dois indivíduos no vetor populacional. Como a população é ordenada (linha 22 no Algoritmo 4), a seleção aleatória é feita apenas com os primeiros 80% da população, o que garante uma variação dos pais da prole (em inglês *offsprings*), para uma melhor caminhada no espaço solução, a fim de reduzir a probabilidade de convergência rápida para um ótimo local e/ou força suficiente para escapar quando este efeito ocorrer. O cruzamento pode acontecer de três maneiras, de acordo com os três pontos aleatórios que são possíveis de serem selecionados e usados como valores de troca para um novo filho. A operação de mutação é uma seleção aleatória com uma probabilidade de 8% para cada filho (linhas 16 a 21). Se uma descendência (novo indivíduo) for selecionada para mutação, um dos três parâmetros da solução atual é selecionado aleatoriamente e recebe um valor aleatório respeitando o intervalo das variáveis. No final do processo de cruzamento, 40 novas soluções são geradas.

A terceira fase é a avaliação da função de ajuste para a descendência gerada. Essa avaliação pode acontecer de duas maneiras. A primeira delas é sem o uso de AM (linhas 22 a 24) e com uma chamada paralela ou serial da simulação para avaliar as 40 novas soluções. A segunda maneira é gerar 120 e 1200 novas soluções, em vez do original 40 e usar o método de AM para prever o valor da descendência. Depois disso, os filhos são classificados e 40 das melhores soluções são simuladas para encontrar seu ajuste (linhas 25 a 28). Como é sabido que a previsão/classificação usando AM pode gerar um erro (diferença entre o valor real e o previsto) (YOUSEFI et al., 2018), diferentes níveis para predição entre 120 e 1200 novos descendentes em cada geração foram testados para verificar se uma previsão em uma solução de 3 até 300 vezes o *crossover* original gera uma solução melhor, respeitando a função de ajuste e o tempo necessário, simulando ainda 40 soluções em cada geração.

A sobrevivência da próxima geração (linhas 29 a 31) é a seleção aleatória de indivíduos entre 10% e 80% relacionada à avaliação simulada (nível de aptidão) e substituída pela nova prole. O próximo passo ao usar uma AM (linhas 32 a 34) é melhorar a base de dados da AM e

acrescentar os valores avaliados e gerados usando a simulação, para um melhor treinamento e previsão para as próximas gerações. A última fase no Algoritmo 2 (linhas 35 a 41) é o tipo de população e conta as interações com melhoria ou não, como os critérios de parada de otimização.

Para a otimização do GRASP, a fase de pré-processamento é considerada no próprio Algoritmo 3, no qual a função de avaliação é criada e aplicada à população. A Tabela 4.10 apresenta o Algoritmo de otimização via GRASP com e sem AM.

Tabela 4.10 - Algoritmo 6 GRASP com e sem AM

Início

```
1   enquanto iter ≤ iterMax faça
2       % Fase 1: Construtiva
3       a, b, c ← listaRestrita(vecPop)
4       offspr1 ← constroiVet(a; b; c)
5       % Phase 2: Construção da solução parcial
6       para cada numbOffspr faça
7           tempVet ← vizinhança(offspr1)
        Fim
        % Fase 3.1: Avaliação da próxima geração sem AM
8       para cada tempVet faça
9           tempRes ← simulAlocaçãoRecursos(tempVet)
        Fim
        % Fase 3.2: Avaliação da próxima geração com AM
10      para cada tempVet faça
11          tempResult ← prevAM(tempVet)
        Fim
        % Phase 3.2.1: Tratamento dos dados para melhoria da AM
12      para cada tempResult(melhor) faça
13          dadosAM ← dadosAM + tempResult(melhor) MOD300
14          marcacoesAM ← marcacoesAM + tempVet
        Fim
        % Fase 4: Encontrar a melhor solução e contagem de iteração
15      starResultAntes ← tempResult(primeiro)
16      vetPop ← ordenaMaxParaMin(tempVet, tempResult)
17      starResultDepois ← tempResult(primeiro)
18      melhorVetList ← melhorVetList + tempVet
19      melhorResList ← bestResList + tempResult
20      se melhorResultDepois ≥ melhorResultAntes faça
21          iter ← 0
        Fim
22      Se
23          iter ← iter + 1
        Fim
    Fim
```

No Algoritmo 6 apresentado na Tabela 4.10, a primeira fase, referente à fase construtiva do procedimento GRASP, é alusiva à elaboração da lista restrita de candidatos para seleção. Para este propósito, um α = 0,8 foi escolhido representando os primeiros 80% dos indivíduos classificados na população inicial gerada no final do Algoritmo 4, sendo candidatos que compõem a lista restrita. Assim, três foram selecionados para gerar uma nova solução com a

primeira variável da primeira seleção com a segunda variável da segunda seleção e assim por diante. Com a solução gerada na fase um, gera-se uma vizinhança que avalia cinco níveis para cada variável (+2, +1, 0, -1, -2), com um total de 125 combinações diferentes ou menos quando um nível atinge a faixa limite de uma variável. Nesse caso, a solução recebe o valor do limite.

A terceira fase é a avaliação da vizinhança, com e sem AM dentro do GA ou GRASP. Quando a avaliação é feita sem a AM, pode ser uma chamada serial ou paralela da simulação. Ao ser feito com AM, a otimização testou dois níveis de 120 a 1200 (10 vizinhanças, ou mais, de cada vez para 10 soluções) para prever e selecionar os melhores 40 e utilizar a função de simulação para avaliação. Isso pode ser considerado um aumento de 4% na pesquisa da solução em cada interação em comparação com a pesquisa do GA. No entanto, neste caso, a probabilidade de gerar soluções na fronteira de alcance compensa esse efeito. Quando a AM é usada em GA ou GRASP, as avaliações são feitas usando a função de paralelismo para geração de soluções. Após a avaliação, a base de dados é atualizada com os valores simulados e a AM é ajustada para uma melhor aprendizagem e previsão na próxima geração. O Algoritmo 5 proposto para o GRASP foi utilizado tanto para os OE1 a 3, de forma que em vez da chamada por $simulAlocRecursos$ da simulação do OE1, a simulação e otimização do OE2 e OE3 chamam respectivamente as funções simulação $simulLotEconomico$ e $simulMaquinasBuffer$.

A quarta fase é a seleção da melhor solução e armazenamento em uma lista para comparação com a melhor solução atual e, finalmente, calcular se alguma melhoria for gerada em comparação com a melhor solução e se os critérios de parada forem atingidos para finalizar a otimização.

Dessa forma, os algoritmos de otimização apresentados no presente capítulo contribuem para a literatura da área de OvS porque possuem dois fatores de ineditismo. O primeiro remete ao fato de que não foram encontrados trabalhos que reunissem os métodos propostos de SED, paralelismo, AM e metaheurística. O segundo fator refere ao caso desta tese ser o primeiro trabalho, até então, que reuniu todas as bibliotecas necessárias na linguagem Python oriundas de outros trabalhos científicos, em um mesmo programa, para se gerar os resultados desejados.

Com o término da construção e implementação dos algoritmos apresentados, a próxima etapa da tese é o teste dos passos aqui apresentados para a verificação da efetividade das teorias apresentadas quanto a utilidade das metaheurísticas, AM e paralelismo para a OvS em modelos de simulação da EP.

5. APLICAÇÃO DO MÉTODO E RESULTADOS

Nesse capítulo são apresentados os resultados das aplicações dos Algoritmos 1 a 6 especificados nas seções 4.3.1 e 4.3.2. Os Algoritmos 3, 4 e 5 geraram os códigos de programação presentes nos Apêndices V, VI e VII. É válido ressaltar que estes códigos formaram as bases para os resultados aqui apresentados. Durante a elaboração de cada etapa da programação, os mesmos foram validados a partir da avaliação dos dados esperados de saída para cada etapa, não apenas o resultado final, gerando assim o processo de validação por partes. A Tabela 5.1 apresenta a relação entre os algoritmos e os respectivos códigos de programação.

Tabela 5.1 - Relação entre algoritmos, implementações e objetivos secundário

Objetivo	**Algoritmo e Código de implementação**
Espaço solução do OE1	Algoritmo 1 e seu código no Apêndice I
Ótimo local do OE2	Algoritmo 2 e seu código no Apêndice II
Espaço solução do OE3	Algoritmo 3 e seu código no Apêndice III
Teste de AM do OE1	
Teste de AM do OE2	Algoritmo 4 e seu código no Apêndice IV
Teste de AM do OE3	
GA + AM + paralelismo no OE1	Algoritmo 4 + 5 e seu código no Apêndice V
GRASP + AM + paralelismo do OE1	Algoritmo 4 + 6 e seu código no Apêndice VI
GRASP + AM + paralelismo do OE2	Algoritmo 4 + 6 e seu código no Apêndice VII
GRASP + AM + paralelismo do OE3	Algoritmo 4 + 6 e seu código no Apêndice VIII

De acordo com a Tabela 5.1, os algoritmos apresentados nas seções 4.3.1 e 4.3.2 foram implementados e seus códigos apresentados nos Apêndices I, II, V, VI e VII. Os testes de AM para o OE1 e OE2 não apresentaram algoritmo específico porque os seus detalhamentos foram expressos de forma discursiva na Seção 4.2.1. Os resultados dos objetivos da Tabela 5.1, com a implementação dos Apêndices I a VII, são apresentados no Capítulo 5.

A escolha de se trabalhar separadamente com a fase de escolha da AM e da OvS foi feita porque no momento de construção e testes dos programas, caso a fase de escolha fosse incorporada, muito tempo seria gasto toda vez que o programa fosse novamente testado, com a incorporação do teste e escolha da AM.

5.1 Geração de soluções de referência do OE1, OE2 e OE3

Como ponto de comparação e métrica para se avaliar o método de OvS aplicado aos objetos de estudo, dois procedimentos foram adotados. Para o OE1 foi gerado todo o espaço solução, e a relação entre o resultado alcançado para a Função Objetivo (FO) e as variáveis

"Pessoal de manutenção" e "Tempo de simulação", as mesmas estão representadas nas Figuras 5.1, 5.2 e 5.3.

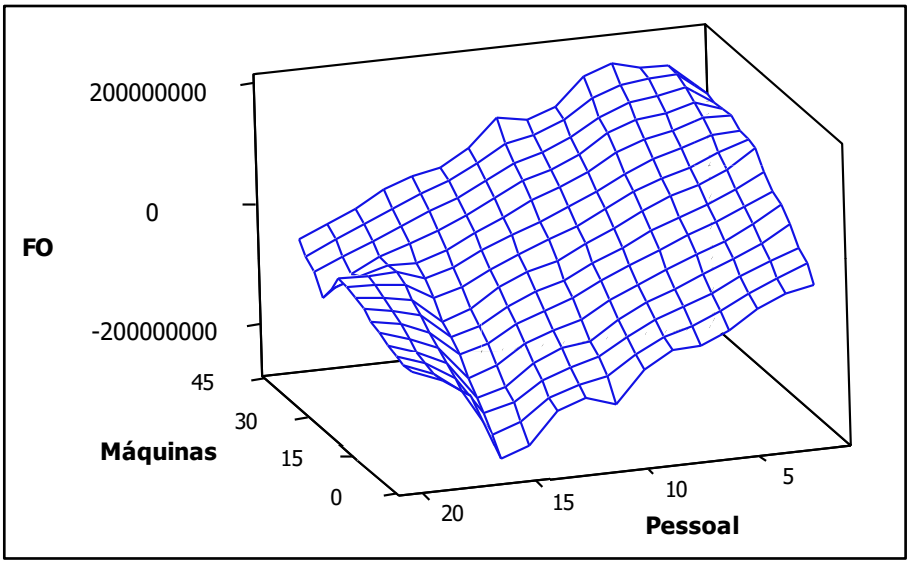

Figura 5.1 - Espaço solução do OE1 para as variáveis FO, máquinas e pessoal

A Figura 5.1 ilustra com o gráfico de superfície relacionando o espaço solução constituído para todas as soluções possíveis para o OE1 referente ao resultado alcançado para a FO e as variáveis Máquinas e Pessoal de Manutenção. A Figura 5.2 faz a relação entre o resultado alcançado para a FO e as variáveis Pessoal de manutenção e Tempo de simulação.

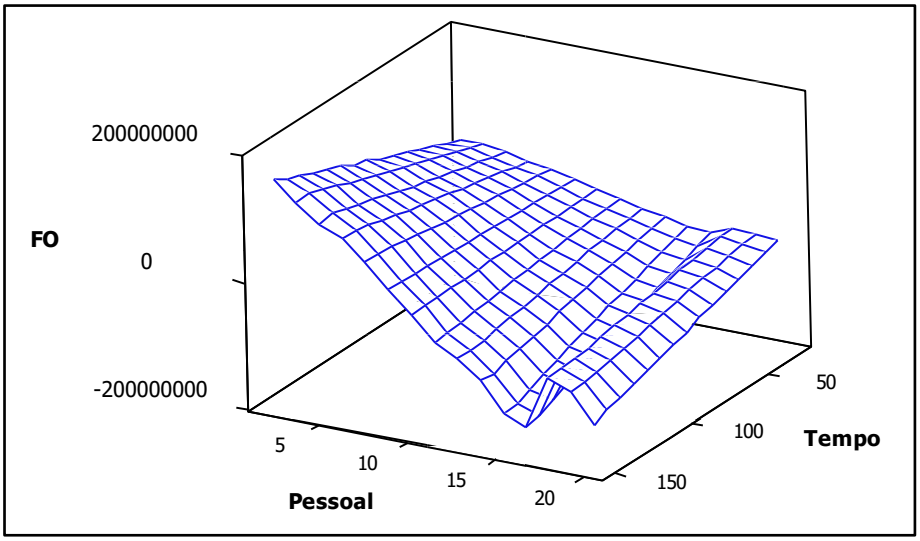

Figura 5.2 - Espaço solução do OE1 para as variáveis FO, pessoal e tempo

De forma análoga, a Figura 5.3 faz referência a interação entre o resultado alcançado para a FO e as variáveis Pessoal de manutenção e Tempo de simulação.

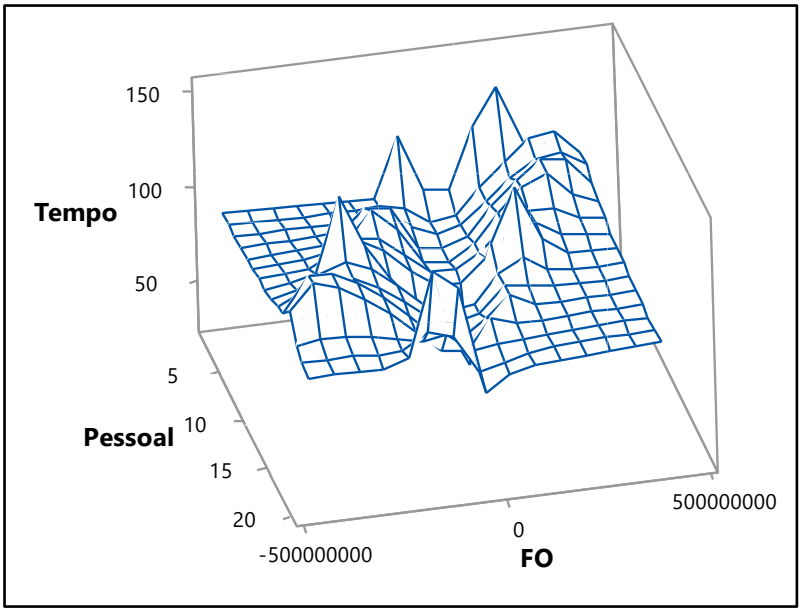

Figura 5.3 - Espaço solução do OE1 para as variáveis tempo, pessoal e FO

Analisando os gráficos das Figuras 5.1, 5.2 e 5.3, é possível perceber que existe uma tendência de relação causa-e-efeito não linear entre as variáveis do problema, de forma que "estruturas" de picos, vales e celas são encontradas percorrendo o espaço solução do problema. A presença desses elementos no espaço solução é uma característica desejável para testes de metaheurísticas, de maneira a oferecer dificuldades ao método de otimização para caminhar no espaço solução e testar a "força" do método em escapar desses pontos e avaliar outras regiões em busca do ótimo global (NOCEDAL; WRIGHT, 2006; RAO, 2009). De forma complementar, as técnicas metaheurísticas são vantajosas de serem aplicadas quando a superfície de resposta para o problema é de alta dimensão, descontínua e não diferenciável (AVCI; SELIM, 2018; TEKIN; SABUNCUOGLU, 2004).

O espaço solução representado nas Figuras 5.1, 5.2 e 5.3 foi gerado de forma serial, demandando um total de 20 dias para encontrar o ótimo global com uma Função Objetivo de FO = 399.473.800 e a solução ótima de $M_i = 40$, $P_j = 4$ e $T_k = 150$, utilizando-se um computador Dell XPS 8910 com 16 GB de RAM equipado com processador Intel i7 7700 a partir do Algoritmo 1 e sua implementação apresentada no Apêndice I. Foi verificado que para o OE1, selecionando 100 soluções distintas e rodando cada uma 10 replicações, em média a diferença entre as replicações foi inferior a 0,03%. Como o OE1 não apresenta o problema de alta variabilidade em suas respostas, foi adotado uma replicação por solução.

Para o OE2, uma busca randômica foi gerada a partir do Algoritmo 2 e da execução do código do Apêndice II. A representação gráfica de parte do espaço solução está representado na Figura 5.4.

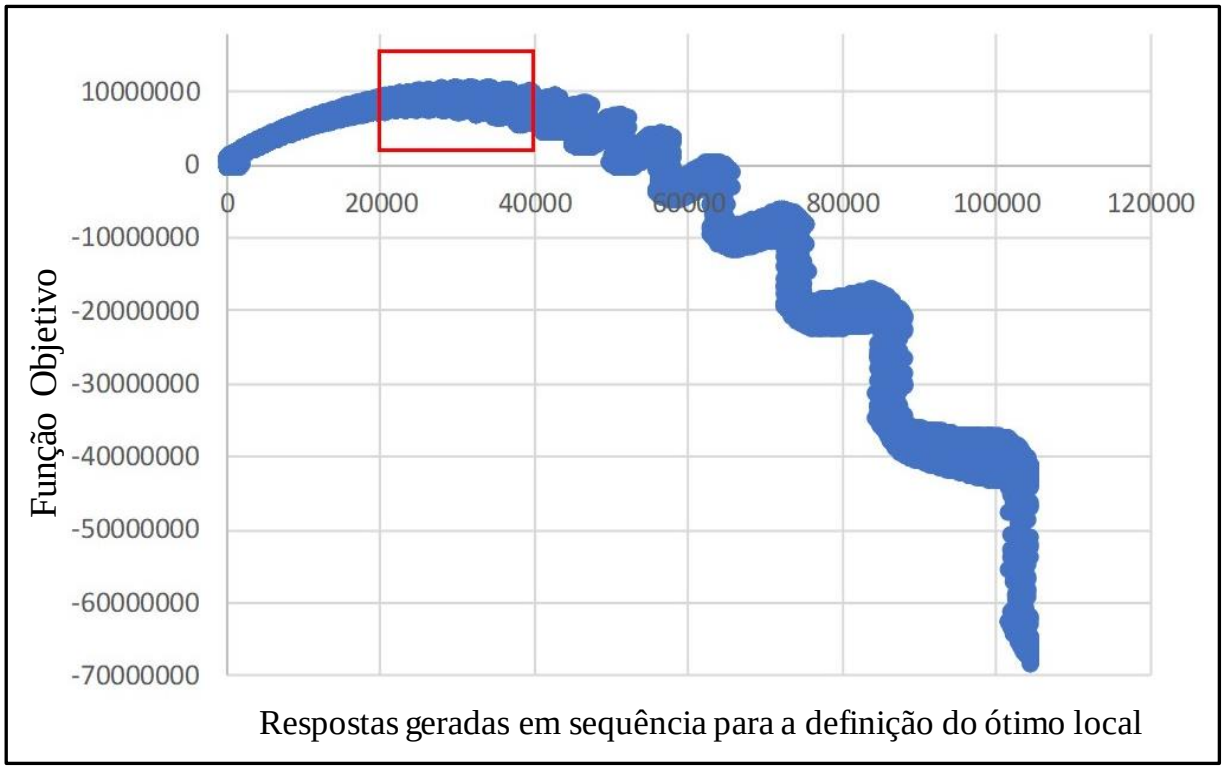

Figura 5. 4 - Parte do espaço solução gerado para o OE2

Para se gerar um efeito parecido com a dos gráficos de superfície apresentados nas Figuras 5.1, 5.2 e 5.3, os dados da Figura 5.4 foram ordenados de maneira crescente para as variáveis lote de reposição, nível de segurança e tempo entre revisões. Dessa forma no ponto (0;0) do eixo "x" estão as respostas com menor valores para as variáveis solução, aumentando com seus valores ao se afastar desse ponto e no eixo "y" é o respectivo valor da FO para a solução. Um total de 242.600 soluções foram geradas em um processamento sequencial ininterrupto durante 13 dias, utilizando o mesmo hardware usado na geração do espaço solução do OE1. Assim como o OE1, para o OE2 foi verificado que 10 replicações por solução testada, é gerado um erro médio de 1,2% para o tempo de espera, erro este aceitável para o problema.

De maneira a melhor identificar visualmente a região onde se encontra o ótimo local, os dados marcados na região em vermelho na Figura 5.4, foram selecionados para a elaboração da Figura 5.5.

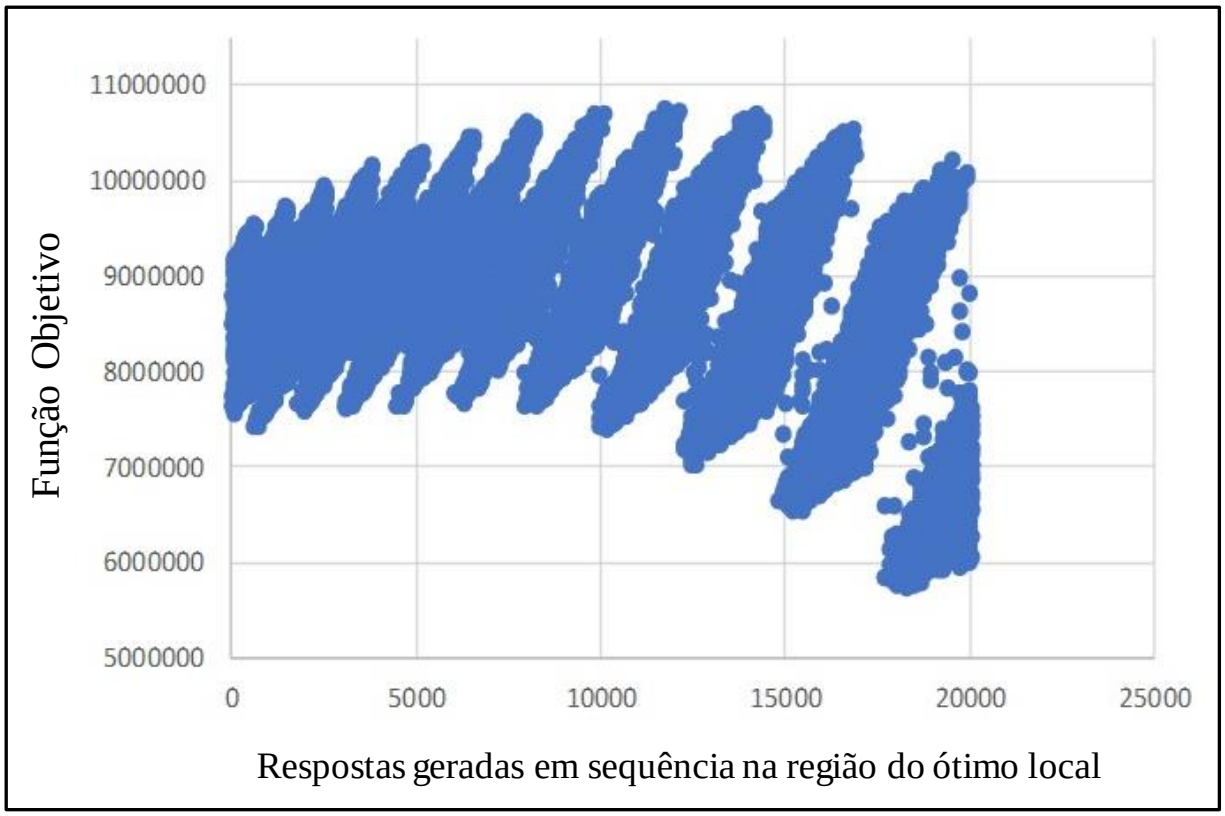

Figura 5.5 - Região do ótimo local para o OE2

Para o OE2 foi identificado que a melhor solução gerada possui FO igual a 10778862 com os parâmetro para lot_i =1570, seg_j =12, rev_k = 2100 e $tempo_{espera}$ = 7930279507 segundos.

Com relação ao OE3, a Figura 5.6 ilustra as 81000 possíveis respostas geradas aleatoriamente sem repetição, ordenadas em ordem crescente para as sete variáveis envolvidas no problema, para representar a dispersão da função objetivo do problema do OE3, com o efeito crescente das variáveis respostas, para se verificar a relação entre o aumento do valor das variáveis da solução com a sua respectiva FO.

Foram utilizadas todas as 81000 possíveis soluções para melhor visualização do gráfico de dispersão de todo o espaço solução, para a visualização do formato do espaço solução.

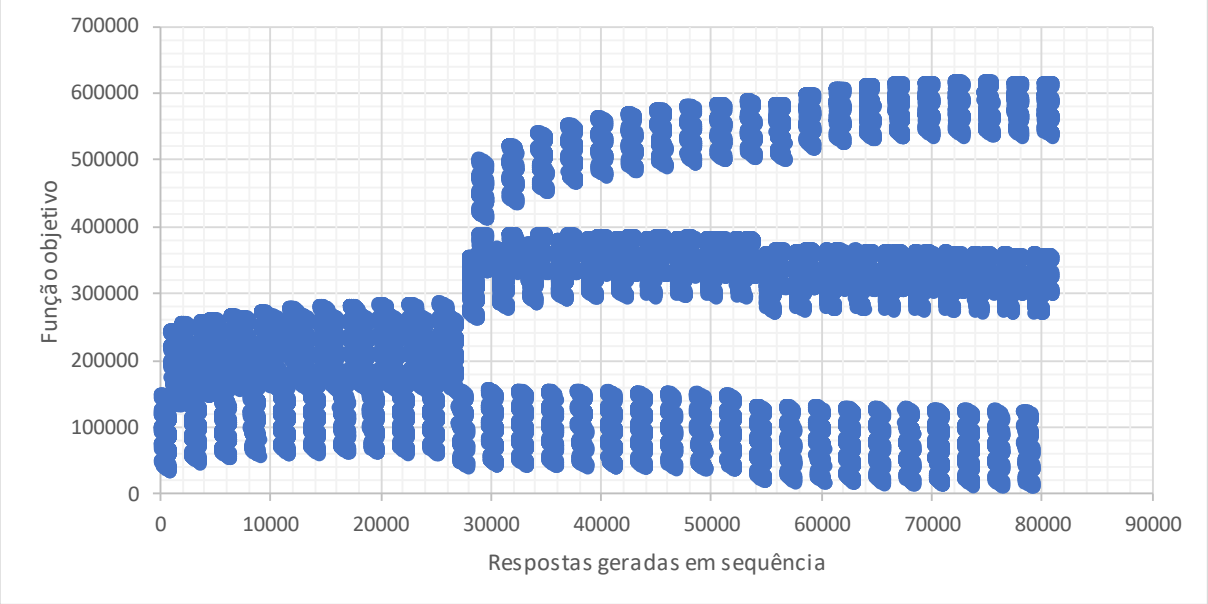

Figura 5.6 - Dispersão das FOs para o espaço solução do OE3

Observando os valores das FOs para o OE3, é possível perceber uma região propensa para se encontrar respostas próximas de um valor ótima, passando por regiões diversas de pontos altos, baixos e de cela, característica esta desejável para teste da OvS proposto. Esta análise visual é confirmada pela geração de todo o espaço solução que possibilitou a determinação do valor ótimo da FO igual a \$618800 para mais de uma resposta, tendo como valores ótimos das variáveis resposta iguais a $M1 = 3$; $B1 = 7$; $M2 = 3$; $B2 = x$; $M3 = 1$; $B3 = 9 - x$; $M4 = 2$, em que o valor de "x" pode variar entre 1 e 8, dessa forma são possíveis 8 combinações de respostas ótimas para o OE3, considerando o conjunto de *software* utilizado.

Analisando as soluções geradas e as soluções ótimas, para o OE1 a resposta ótima representa valores de fronteira para a amplitude das variáveis de resposta. Para o OE2 e OE3, a resposta ótima representa valores médios para as variáveis de resposta. Dessa maneira, os objetos de estudo oferecem o desafio de encontrar valores nas fronteiras ou dentro dos espaços soluções, característica esta desejável como estudo de diferentes comportamentos para teste do método de OvS proposto pelo presente trabalho.

O passo seguinte para o teste do método é a avaliação da AM para os objetos de estudos é verificar efetivamente a contribuição do uso dessas técnicas de previsão/classificação.

5.2. Geração da base de dados e teste das AM

Nessa seção é testada a utilização da AM para se avaliar como a mesma pode ser aplicada de forma útil para a proposta de OvS. Dessa maneira, foi avaliada a forma como a AM

deve ser utilizada e qual a melhor AM que deve ser selecionada para os testes de otimização nos objetos de estudo. Como índice de referência foram avaliados os valores do MAE, R^2 e IC (95%), os índices Min, Max e MSE foram utilizados após o tratamento dos bancos de dados. Em ambos os casos foi adotado o valor de MAE para a ordenação do melhor método, assim como explicado na Seção 4.2.1.

5.2.1 Teste piloto de treino e previsão para os objetos de estudo

Para se avaliar o desempenho da AM nos objetos de estudo, os 33 métodos foram aplicados nos OE1, 2 e 3 seguindo os passos apresentados na Seção 4.2.1. A Tabela 5.2 apresenta os resultados para o primeiro teste com o OE1.

Tabela 5.2 - AM aplicado ao OE1

AM	MAE	R^2	T – Treino (s)	T – Prev. (s)	IC (95%)
PR	265.627.744.175	0,86	<0,00	<0,00	(-300.811,08 ; 10.985.066,85)
MLPR	285.202.004.506	0,84	0,13	<0,00	(-1.081.069,06 ; 10.848.622,46)
EM	298.030.723.085	0,84	<0,00	<0,00	(-1.163.401,23 ; 10.976.595,55)
KRR	303.079.308.215	0,83	0,18	<0,00	(-1.914.518,82 ; 10.250.232,11)
RRCV	298.114.391.191	0,83	<0,00	<0,00	(-1.167.263,53 ; 10.977.217,34)
RR	298.136.842.933	0,83	<0,00	<0,00	(-1.168.101,81 ; 10.977.355,11)
LASSO	298.138.024.843	0,83	<0,00	<0,00	(-1.168.145,97 ; 10.977.362,40)
OLS	298.138.024.872	0,83	<0,00	<0,00	(-1.168.145,97 ; 10.977.362,40)
OMP	298.138.024.872	0,83	<0,00	<0,00	(-1.168.145,97 ; 10.977.362,40)
GTBR	334.696.730.576	0,82	0,02	<0,00	(-2.160.255,73 ; 10.578.121,25)
ETC	288.603.530.000	0,79	0,63	0,02	(5.388.452,72 ; 18.684.789,28)
DTR	283.095.495.000	0,74	<0,00	<0,00	(-3.747.388,76 ; 11.542.717,76)
DTC	346.898.735.000	0,73	0,16	<0,00	(-7.326.360,32 ; 8.157.277,32)
RFC	359.154.415.000	0,71	0,68	0,02	(12.757.056,46 ; 27.854.220,54)
SVM	341.088.550.000	0,71	53078,00	44371,00	(-8.991.592,12 ; 7.160.958,12)
LR	310.634.985.000	0,70	62204,00	<0,00	(-3.164.552,34 ; 13.202.781,34)
GNB	349.229.985.000	0,70	0,07	0,04	(-2.886.295,79 ; 13.480.302,79)
NCC	349.229.985.000	0,70	0,04	<0,00	(-2.886.295,79 ; 13.480.302,79)
MNB	408.912.655.000	0,67	0,03	<0,00	(-16.601.313,53 ; 423.226,53)
RFR	508.347.681.908	0,65	0,01	<0,00	(-5.720.321,42 ; 11.926.234,22)
GPR	482.511.212.485	0,56	178057,00	0,01	(-6.719.530,38 ; 13.309.509,69)
MLPC	520.492.425.000	0,54	327937,00	0,02	(-16.638.368,18 ; 4.372.953,18)
OVOC	554.737.840.000	0,51	11066904,00	885783,00	(-55.299.256,15 ; -38.328.399,85)
SGD	635.787.485.000	0,49	10662,00	<0,00	(-10.078.943,03 ; 14.500.834,03)
BRR	831.673.867.904	0,33	<0,00	<0,00	(-7.741.427,79 ; 22.333.807,34)
ARDR	831.673.867.904	<0,00	20439,00	<0,00	(-7.741.427,79 ; 22.333.807,34)
NNC	900.105.435.000	<0,00	<0,00	0,02	(80.823.521,61 ; 97.997.539,39)
ABC	886.197.350.000	<0,00	78311,00	0,87	(-10.085.660,46 ; 20.763.148,46)
ECOC	1.093.781.235.000	<0,00	2358537,00	0,09	(8.583.913,10 ; 47.086.803,90)

AM	MAE	R^2	T – Treino (s)	T – Prev. (s)	IC (95%)
OVRC	1.193.686.260.000	<0,00	31296,00	0,03	(22.048.736,22 ; 62.129.199,78)
BC	1.320.755.110.000	<0,00	0,01	0,03	(53.960.473,63 ; 98.708.450,37)
BNB	4.473.677.835.000	<0,00	0,03	<0,00	(432.059.352,15 ; 462.676.214,85)
LDA	----------	-----	----------	----------	----------

Para se ordenar os dados da Tabela 5.2 foi utilizado como critério a média do erro absoluto (MAE) nesse teste inicial da AM, do menor para o maior valor, com exceção do método LDA que, ao ser aplicado, não gerou dados para essa base de dados. Dessa forma, o melhor método foi o PR seguido pelo MLPR. É válido ressaltar que a magnitude dos erros associados às previsões/classificações é inferior à magnitude dos valores das respostas. A Tabela 5.3 apresenta os valores para a aplicação da AM para o OE2.

Tabela 5.3 – AM aplicado ao OE2

AM	MAE	R2	T-Treino (s)	T-Prev (s)	IC (95%)
DTR	2.784.277.650	0,99	0,01	<0,00	(-187.911,43 ; -2.064,14)
GTBR	11.133.700.843	0,99	0,02	<0,00	(-285.203,72 ; 244.730,37)
PR	11.432.114.031	0,99	0,01	<0,00	(-340.695,17 ; 154.414,38)
MNB	12.461.142.050	0,97	0,03	<0,00	(-854.676,13 ; 17.708,84)
ETC	27.146.846.400	0,86	0,68	0,02	(675.619,68 ; 2663.242,34)
LR	27.794.480.550	0,92	915.082,00	<0,00	(-318.523,70 ; 1280.773,77)
RFR	28.803.631.444	0,94	0,01	<0,00	(-887.854,72 ; 426.939,94)
RFC	61.112.252.250	0,67	15.049,00	0,02	(2.056.102,40 ; 5071.639,81)
EM	71.057.172.001	0,83	0,02	<0,00	(-2.313.943,81 ; -53.234,01)
RR	71.057.172.973	0,83	<0,00	<0,00	(-2.313.943,49 ; -53.233,65)
LASSO	71.057.172.982	0,83	0,03	<0,00	(-2.313.943,49 ; -53.233,65)
OLS	71.057.172.983	0,83	<0,00	<0,00	(-2.313.943,49 ; -53.233,65)
OMP	71.057.205.438	0,83	<0,00	<0,00	(-2.321.129,51 ; -61.005,59)
KRR	76.095.067.550	0,79	15.942,00	<0,00	(-3.791.657,61 ; -1.377.207,36)
ARDR	80.180.630.421	0,77	287.369,00	<0,00	(-2.681.638,56 ; -43.920,32)
DTC	131.737.038.500	0,09	13.474,00	<0,00	(-5.226.884,42 ; 527.390,62)
BC	137.269.781.400	0,16	0,01	0,03	(5.56.863,28 ; 6.451.312,62)
ABC	149.656.217.850	0,24	85.770,00	0,84	(-13.884.691,36 ; -8.560.133,47)
MLPC	155.179.537.250	0,24	48.750,00	0,02	(-12.589.829,20 ; -7.039.269,13)
OVRC	155.196.081.650	0,24	20.804,00	0,03	(-12.534.681,20 ; -6.984.121,13)
GPR	155.626.865.750	0,20	13.889,00	0,02	(-11.682.088,20 ; -6.131.528,13)
GNB	163.789.130.250	0,37	0,07	0,04	(-3.345.682,01 ; 3.143.821,04)
NCC	163.789.130.250	0,37	0,04	<0,00	(-3.345.682,01 ; 3.143.821,04)
BRR	165.926.853.902	0,01	<0,00	<0,00	(-4.677.660,71 ; 878.821,77)
MLPR	166.088.122.084	0,01	0,01	<0,00	(-4.645.587,90 ; 908.503,89)
SGD	178.611.031.250	<0,00	0,63	<0,00	(-20.542.957,20 ; -14.992.397,13)
ECOC	182.710.415.050	<0,00	2.422.322,00	0,09	(-21.007.260,20 ; -15.456.700,13)
SVM	193.633.921.650	<0,00	54.863,00	33590,00	(-22.138.672,20 ; -16.588.112,13)

AM	MAE	R2	T-Treino (s)	T-Prev (s)	IC (95%)
NNC	313.310.696.950	<0,00	<0,00	0,02	(27.524.883,78 ; 33.088.599,47)
BNB	577.854.580.350	<0,00	0,03	<0,00	(55.003.575,80 ; 60.554.135,87)
RRCV	270.851.175.837.144	<0,00	<0,00	<0,00	(-5.554.497.703,44 ; 4.108.639.165,20)
OVOC	----------	------	----------	----------	----------
LDA	----------	------	----------	----------	----------

De acordo com os dados presentes na Tabela 5.3, para o OE2 os melhores métodos de AM estão dispostos de forma decrescente, de maneira que os métodos DTR e GTBR representam os melhores métodos para previsão da resposta do problema. Assim como no OE1, ocorreu dos métodos LDA e OVOC não conseguirem gerar dados, devido a erro de processamento da base de dados do OE2. Na Tabela 5.4 é apresentado os resultados para a aplicação dos 33 MLs para o OE3.

Tabela 5.4 – AM aplicado ao OE3

AM	MAE	R2	T-Treino (s)	T-Prev (s)	IC (95%)
DTR	2526,37	0,99	0,01	<0,00	(-789,55 ; 20,88)
ETC	6585,90	0,98	10281,00	0,05	(-1054,42 ; 3725,96)
DTC	7093,47	0,99	0,17	<0,00	(-1565,59 ; 899,28)
LDA	8216,19	0,98	0,23	0,01	(-2981,51 ; 1513,10)
RFC	11710,71	0,95	0,97	0,05	(3951,71 ; 10536,01)
GNB	13961,36	0,96	0,13	0,10	(-6536,20 ; -277,38)
PR	14913,97	0,99	0,21	<0,00	(-1369,43 ; 2211,41)
GPR	18750,38	0,97	510599,00	0,05	(2683,93 ; 8327,51)
LR	23703,39	0,87	357200,00	0,01	(-9353,25 ; 1679,09)
MLPC	54150,39	0,59	572503,00	0,01	(-8411,62 ; 11609,53)
MLPR	54705,19	0,80	0,26	<0,00	(-2562,84 ; 11220,87)
ABC	55110,71	0,66	173920,00	18141,00	(21458,79 ; 38650,87)
RFR	55403,28	0,78	0,05	<0,00	(-8871,99 ; 5749,43)
GTBR	58777,38	0,77	0,04	<0,00	(1011,25 ; 15828,86)
KRR	68463,98	0,71	10111,00	0,01	(-5463,18 ; 11443,91)
EN	68479,73	0,73	<0,00	<0,00	(-645,48 ; 15673,86)
ARDR	68936,65	0,73	3105358,00	<0,00	(-565,79 ; 15734,30)
BRR	68954,16	0,73	<0,00	<0,00	(-513,09 ; 15787,54)
RRCV	68958,71	0,73	<0,00	<0,00	(-512,01 ; 15788,82)
RR	68961,15	0,73	0,03	<0,00	(-511,44 ; 15789,51)
LASSO	68963,56	0,73	<0,00	<0,00	(-510,86 ; 15790,20)
OLS	68963,59	0,73	<0,00	<0,00	(-510,86 ; 15790,20)
OMP	69061,29	0,72	<0,00	<0,00	(-1361,58 ; 15088,58)
NCC	76034,99	0,40	0,07	0,01	(-19983,27 ; 4189,01)
SGD	85401,57	0,35	43452,00	0,02	(-16400,63 ; 8883,14)
MNB	86631,85	0,26	0,11	0,01	(33373,90 ; 58557,17)
SVM	103740,99	0,03	63164,00	106686,00	(62314,45 ; 89152,91)

AM	MAE	R2	T-Treino (s)	T-Prev (s)	IC (95%)
ECOC	103832,38	0,05	6820480,00	0,20	(70811,76 ; 96284,85)
NNC	121675,72	0,20	0,21	0,09	(79200,18 ; 107821,23)
BC	148703,92	0,23	0,03	0,11	(29425,48 ; 69954,16)
BNB	198350,39	0,15	0,10	0,01	(178649,03 ; 209907,62)
OVRC	287344,13	0,03	425664,00	0,07	(-294964,14 ; -260402,96)
OVOC	----------	----------	----------	----------	----------

Analisando a Tabela 5.4, o método que gerou melhor ajuste ao banco de dados do OE3 foi a AM DTR, com melhor valor referente ao MAE, e tempos de treinamento e previsões inferiores a um segundo, tornando este um bom método candidato para ser utilizado como metamodelo do OE3.

Com os testes iniciais apresentados pelos resultados das Tabelas 5.2, 5.3 e 5.4 foi possível verificar duas características importantes. A primeira característica remete a testar a possibilidade de se utilizar AM para os objetos de estudo, de forma que a mesma possa ser empregada como um metamodelo, substituindo com certa precisão a necessidade de utilização da SED. A segunda característica se refere à utilização de um metamodelo ao invés da SED para a solução desejada.

Nos 3 objetos de estudo, os melhores métodos de AM demandaram um tempo de treinamento e previsão menor que um segundo, importante resultado para o presente trabalho pois com a AM, 400 previsões foram geradas com tempos inferior a 0,09 segundo, em sua maioria, ao passo que a simulação de apenas uma solução via SED demanda, em média, mais de um segundo. Assim para os testes experimentais, existem métodos de AM que tiveram um desempenho 400 vezes melhor comparado com a utilização da SED, com um intervalo de confiança associado ao erro do mesmo.

Portanto para este teste piloto de aplicação de AM pode se avaliar que a utilização da mesma gera resultados mais rapidamente, porém possui um erro associado, então o seu emprego apenas, não é 100% confiável para se encontrar um ótimo local. A AM foi 400 vezes mais rápida que a simulação, porém associada a outro método de otimização como as metaheurísticas, não se sabe qual o real ganho de desempenho e se o mesmo será multiplicado ou reduzido quando da associação com a metaheurística ou outro método/ ferramenta de otimização como o paralelismo.

5.2.2 Novo teste das AM e seleção para otimização

Na Seção 5.2.1 foram aplicadas as AMs diretamente nas bases de dados dos objetos de estudo. Analisando os dados de saída gerados pela FO das simulações, estes são elementos de

ordem contínua (não discreta) e com grande variabilidade. Para dados contínuos é recomendado a utilização de AMs para previsão. No intuito de permitir a utilização de uma grande variabilidade de AMs, os valores das FOs foram transformados no primeiro valor inteiro do quociente da divisão por 300000 (OE1 e OE2) e 30 (OE3), de acordo com a Equação (16).

$$FO' = \left\lceil \frac{FO}{300000} \right\rceil \tag{16}$$

A escolha do valor 300000 é determinado pela consideração de que uma diferença de menos de 300000 entre os resultados tem o mesmo valor do ponto de vista da tomada de decisão para os OE1 e OE2, sendo utilizado o valor 30 para o OE3, além de serem divisores que geram números maiores que zero para todas as respostas. Essa transformação de dados é necessária para a criação de classes e geração de uma base de dados que possibilita o uso e ajuste de mais de uma variedade de técnicas de AM (KUBAT, 2017). Na Tabela 5.5 é apresentado o resultado da utilização das AMs para o OE1, a partir das classes geradas pelos múltiplos de 300000, ordenados pelo menor valor do MAE.

Tabela 5.5 - AM aplicados a nova base de dados do OE1

Método	**Min**	**Max**	**MAE**	**MSE**	**R^2**	**T - Trein. (s)**	**T - Prev. (s)**	**IC (95%)**
GPR	-5,3	14,7	0,7	2,1	0,99	11,89	0,01	(-0,19 ; 0,21)
PR	-15,2	19,8	2,5	15,2	0,99	<0,00	<0,00	(-0,29 ; 0,80)
DTR	-21,0	25,0	3,1	25,3	0,99	<0,00	<0,00	(-1,07 ; 0,29)
RFC	-18,0	25,0	3,4	29,9	0,98	0,08	<0,00	(-0,11 ; 1,50)
ETC	-26,0	42,0	4,3	49,8	0,97	0,08	<0,00	(-0,16 ; 1,79)
DTC	-28,0	27,0	4,7	53,7	0,97	0,01	<0,00	(-0,81 ; 1,11)
MLPR	-19,1	40,5	5,9	69,6	0,96	0,12	<0,00	(-0,47 ; 1,86)
GNB	-40,0	42,0	11,0	206,9	0,90	0,01	0,01	(-1,79 ; 2,23)
LDA	-52,0	57,0	11,3	229,9	0,88	0,04	<0,00	(-2,20 ; 2,03)
GTBR	-56,7	57,5	13,1	339,1	0,83	0,02	<0,00	(-2,88 ; 2,27)
ARDR	-57,0	55,1	13,1	353,1	0,82	3,53	<0,00	(-2,56 ; 2,69)
EN	-57,1	55,4	13,1	353,1	0,82	<0,00	<0,00	(-2,57 ; 2,68)
LASSO	-57,1	55,4	13,1	353,3	0,82	<0,00	<0,00	(-2,58 ; 2,68)
BRR	-57,2	55,4	13,1	353,3	0,82	<0,00	<0,00	(-2,58 ; 2,68)
RRCV	-57,2	55,3	13,1	353,4	0,82	0,01	<0,00	(-2,58 ; 2,68)
RR	-57,2	55,3	13,1	353,4	0,82	0,03	<0,00	(-2,58 ; 2,68)
OLS	-57,2	55,3	13,1	353,4	0,82	0,03	<0,00	(-2,58 ; 2,68)
OMP	-57,2	55,3	13,1	353,4	0,82	<0,00	<0,00	(-2,58 ; 2,68)
KRR	-58,5	56,5	13,1	358,1	0,82	0,17	<0,00	(-3,05 ; 2,23)
NNC	-82,0	57,0	15,2	422,8	0,79	<0,00	0,01	(4,79 ; 10,14)
LR	-49,0	57,0	17,3	483,0	0,76	0,65	<0,00	(-2,78 ; 3,49)

Método	Min	Max	MAE	MSE	R^2	T - Trein. (s)	T - Prev. (s)	IC (95%)
RFR	-75,9	90,0	17,9	581,8	0,71	0,01	<0,00	(-3,19 ; 3,56)
MNB	-73,0	80,0	19,1	612,7	0,69	0,01	<0,00	(1,68 ; 8,46)
OVOC	-73,0	105,0	17,7	661,4	0,67	22,01	1,36	(-0,37 ; 6,76)
NCC	-141,0	122,0	23,9	1217,3	0,39	0,01	<0,00	(-8,08 ; 1,64)
SGD	-118,0	113,0	28,6	1467,7	0,26	0,12	<0,00	(-6,05 ; 6,55)
ABC	-118,0	135,0	28,5	1475,8	0,26	0,98	0,05	(-3,82 ; 6,91)
ECOC	-113,0	111,0	30,9	1518,3	0,23	31,37	0,01	(-6,58 ; 4,30)
MLPC	-123,0	135,0	33,9	1908,5	0,04	3,23	<0,00	(-4,94 ; 7,27)
BNB	-137,0	135,0	34,7	1983,0	<0,00	0,01	<0,00	(-6,34 ; 6,10)
SVM	-158,0	114,0	35,9	2316,2	<0,00	0,57	0,15	(-23,92 ; -11,40)
BC	-124,0	192,0	40,0	2909,7	<0,00	0,01	0,01	(6,47 ; 19,31)
OVRC	-98,0	186,0	58,9	4776,9	<0,00	2,26	<0,00	(39,15 ; 53,49)

Analisando os dados da Tabela 5.5, os índices Min, Max, MAE e MSE apresentam valores em classes (múltiplos de 300000), os quais os melhores métodos considerados, de acordo com o menor índice MSE e MAE, foram o GPR e PR para o OE1. Desta forma, estes métodos foram selecionados para serem aplicados à OvS do OE1.

Os únicos parâmetros em que o GPR perde para o segundo método AM (PR) são os tempos necessários para treinamento e previsão. Essas perdas demonstram uma margem significativa. Este fato pode ser um problema porque uma AM pode gerar melhores soluções, mas pode não ser preferível, considerando que é apenas um sistema de suporte que passará por um segundo processo (simulação real) para uma avaliação final. Por esse motivo, duas técnicas de AM (GPR e PR) foram selecionadas para avaliação da melhor combinação de métodos nas duas técnicas de otimização selecionadas (GA e GRASP).

A Figura 5.7 ilustra o *boxplot* definido para os dados do erro nos primeiros nove métodos apresentados na Tabela 5.5. A escolha de mostrar apenas os nove primeiros se deve pelo fato de que a partir do décimo método, os dados geram valores com escala muito ampla, dificultando a análise dos principais métodos.

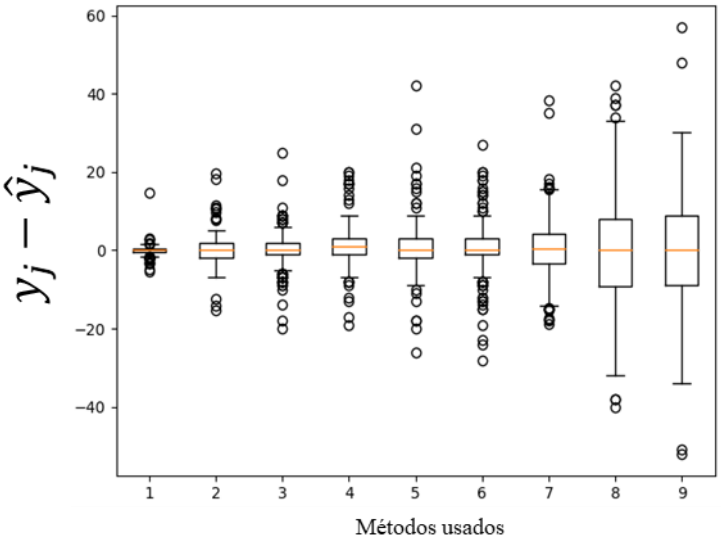

Figura 5.7 - *Boxplots* para o erro das nove primeiras AM no OE1

Analisando a Figura 5.7, o primeiro conjunto de *boxplot* mostra que quanto melhor a previsão da AM, mais próximo o erro está de ser o valor zero. Isso é um bom sinal de que a AM selecionada para otimização (GPR e PR) tem a capacidade desejável de fazer boas previsões com um erro esperado próximo ao valor real.

A Figura 5.8 apresenta os *boxplots* que compara a distribuição dos dados da previsão das nove melhores AM da Tabela 5.4, com o *boxplot* apresentando no número "1" referente aos valores reais que devem ser previstos.

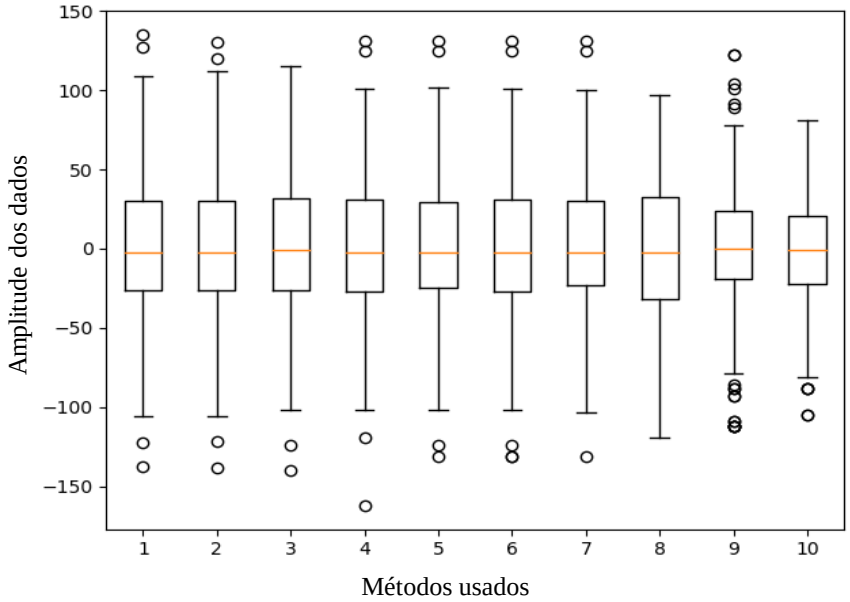

Figura 5.8 - Dispersão dos dados para os nove primeiros métodos

Analisando os dados presentes na Figura 5.8, para métodos com erros de previsão próximos um do outro, a dispersão dos dados possui uma distribuição muito próxima uma das outras, com diferenças perceptíveis e consideráveis quando o método começa a se posicionar em colocação "distante" dos melhores métodos. Dessa forma, a análise da dispersão dos dados não auxilia na seleção de um método, mas pode ajudar na caracterização se o mesmo tem melhor ou pior chance de ser aplicado.

Cabe fazer a comparação entre os dados presentes nas Tabelas 5.2 e 5.5, de maneira tal que com a transformação dos dados e a respectiva redução da variabilidade, o erro associado às previsões e classificações diminuiu. Ocorreu a redução dos valores do IC (95%), tendo como referência as classes para transformação dos dados, e o valor do R^2 aumentou para vários métodos, acima de 90%, valor este não encontrado na Tabela 5.2.

A Tabela 5.6 apresenta os resultados para a transformação da base de dados do OE2 e a aplicação dos métodos de AM, ordenados pelos menores valores do MAE.

Tabela 5.6 - AM aplicados a nova base de dados do OE2

Método	**Min**	**Max**	**MAE**	**MSE**	**R2**	**T - Treino (s)**	**T - Prev. (s)**	**IC (95%)**
DTR	-350000	4830000	76400	9047867	0,99	0,02	<0,00	(-1,91 ; 4,93)
DTC	-6550000	4100000	141300	28649633	0,98	0,19	<0,00	(-7,47 ; 4,71)
ETC	-410000	7430000	194367	57637833	0,98	0,83	0,03	(6,32 ; 23,27)
RFC	-9750000	9390000	225033	80217967	0,98	0,92	0,02	(-1,08 ; 19,20)
GTBR	-3605023	1561735	292795	27074264	0,99	0,05	<0,00	(-11,41 ; 0,37)
PR	-3065231	1086772	337690	24519506	0,99	0,02	<0,00	(-5,01 ; 6,26)
LDA	-4080000	5910000	518733	85233667	0,98	0,13	<0,00	(4,42 ; 25,16)
MNB	-6440000	6550000	724233	144621367	0,96	0,05	<0,00	(26,78 ; 52,62)
RFR	-9096776	3242486	872532	161065033	0,95	0,02	<0,00	(-9,38 ; 19,49)
LR	-10680000	14480000	1485167	706014433	0,80	862K	0,02	(-59,41 ; 0,70)
OMP	-6748712	3992626	2166732	627046043	0,82	<0,00	<0,00	(-30,47 ; 26,52)
BRR	-6776261	3982211	2167095	627438510	0,82	<0,00	<0,00	(-30,60 ; 26,41)
LASSO	-6789815	3976429	2167569	627923027	0,82	<0,00	<0,00	(-30,71 ; 26,32)
EM	-6789821	3976426	2167570	627923213	0,82	<0,00	<0,00	(-30,71 ; 26,32)
OLS	-6789840	3976419	2167570	627923830	0,82	<0,00	<0,00	(-30,71 ; 26,32)
RR	-6789840	3976419	2167570	627923830	0,82	<0,00	<0,00	(-30,71 ; 26,32)
KRR	-8840000	3470000	2324412	746439534	0,79	88K	<0,00	(-45,94 ; 16,16)
ARDR	-11054335	4496943	2516239	905193249	0,74	818K	<0,00	(-36,30 ; 32,18)
ABC	-14900000	7870000	2872700	1541992433	0,56	95K	10K	(13,85 ; 102,25)
SGD	-23360000	2620000	4367400	4290644800	0,24	23K	<0,00	(-353,24 ; -219,12)
OVRC	-22820000	3160000	4378200	4010730400	0,16	120K	0,03	(-299,24 ; -165,12)
ECOC	-22620000	3360000	4396200	3921858400	0,13	2653K	0,15	(-279,24 ; -145,12)
GPR	-22530000	3450000	4405333	3884476000	0,12	62K	0,08	(-270,24 ; -136,12)
GNB	-16890000	15140000	4500133	4509080333	0,30	0,08	0,05	(-201,44 ; -51,32)

Método	Min	Max	MAE	MSE	R2	T - Treino (s)	T - Prev. (s)	IC (95%)
MLPR	-20652886	5955461	4898034	3478535867	<0,00	0,03	<0,00	(-38,51 ; 95,58)
BNB	-25480000	500000	5023667	5953488000	<0,00	0,03	<0,00	(-565,24 ; -431,12)
SVN	-25480000	500000	5023667	5953488000	<0,00	34K	27K	(-565,24 ; -431,12)
BC	-21970000	19430000	5133967	5404094233	<0,00	0,02	0,06	(-67,56 ; 99,73)
MLPC	-18860000	7120000	5446733	3740024800	<0,00	57K	0,02	(96,76 ; 230,88)
NCC	-16850000	21960000	5904533	6556824600	<0,00	0,03	0,02	(139,46 ; 316,32)
NNC	-23110000	22380000	8371800	$11269*10^6$	<0,00	<0,00	0,03	(494,79 ; 694,92)
RRCV	$-275*10^6$	$398*10^6$	$83*10^6$	$1212*10^{15}$	<0,00	<0,00	<0,00	(-955451,92 ; 1549775,95)
OVOC	-----	-----	-----	-----	-----	-----	-----	-----

De acordo com a Tabela 5.6, vários métodos de AM foram aplicados com uma boa aderência aos dados reais (R^2>0,90), sendo eleito com menor valor de MAE e MSE, o método DTR para aplicação na OvS do OE2. É interessante ressaltar que o método de classificação DTC na Tabela 5.3 estava na posição 16 passou para a segunda posição, podendo ser essa inversão de ordem considerada uma evidência de que a transformação da base de dados possibilitou um melhor ponto de comparação para utilização de diferentes tipos de métodos.

A utilização da AM com a nova base de dados gerada pela transformação das FOs gerou uma melhora nos índices MAE, R^2 e IC, aumentando a quantidade de métodos de AM com melhor aderência aos dados, por exemplo, o método DTC para o OE2 que possuía R^2 igual a 0,09 e passou para 0,98, fazendo uma comparação direta.

A Figura 5.9 ilustra o *boxplot* definido para os dados do erro nos primeiros 12 métodos apresentados na Tabela 5.5. A escolha de mostrar apenas os 12 primeiros se deve ao fato de que a partir do décimo segundo método, os dados geram valores que aumentam a escala do gráfico e dificulta a análise dos principais métodos.

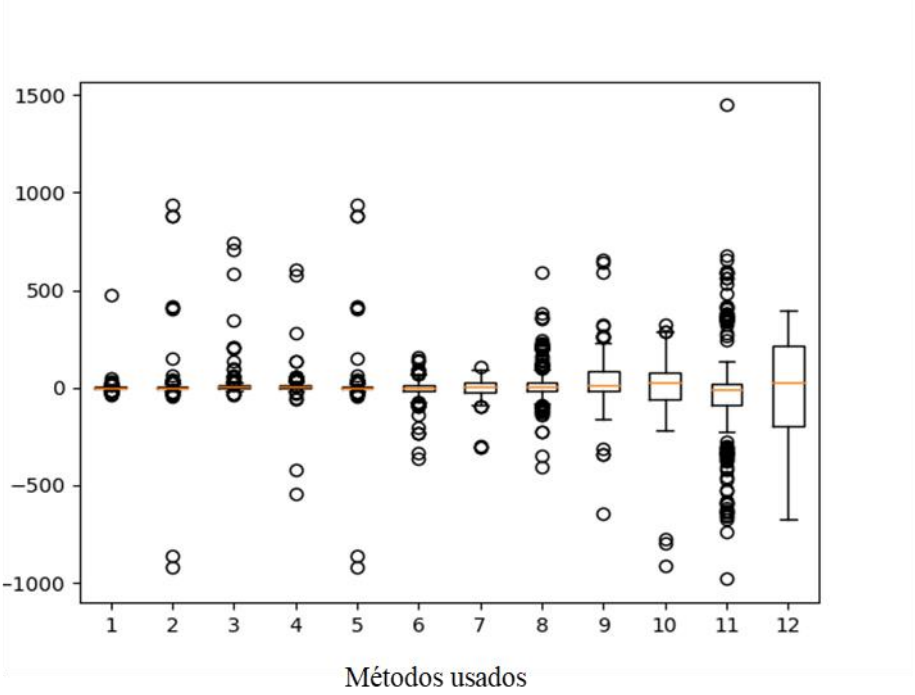

Figura 5.9 - *Boxplots* para o erro das 12 primeiras AM no OE2

A Figura 5.10 apresenta os *boxplots* que compara a distribuição dos dados da previsão das 12 melhores AM da Tabela 5.6, com o *boxplot* apresentando no número "1" referente aos valores reais que devem ser previstos pelo melhor método da Tabela 5.6, e assim por diante.

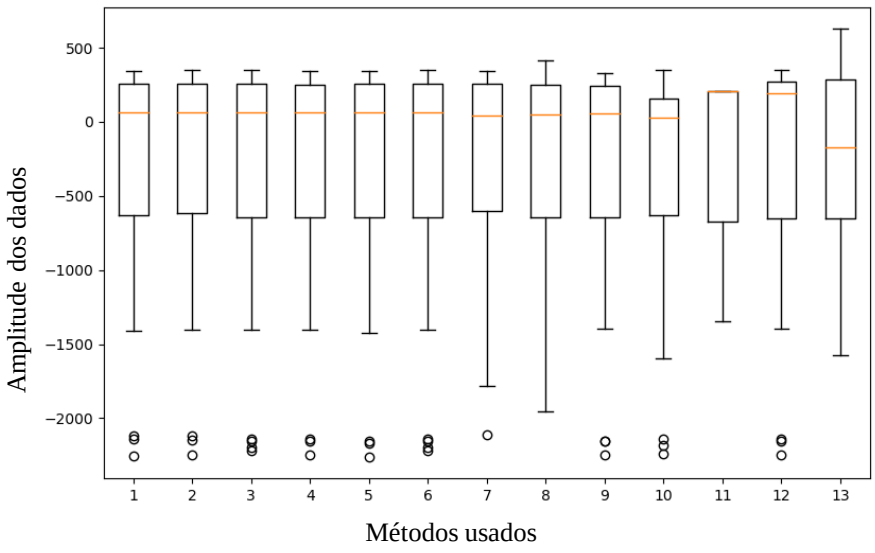

Figura 5.10 - Dispersão dos dados para os 12 primeiros métodos

Comparando a Figura 5.10 com a Figura 5.8, a dispersão dos dados não foi um fator crítico que contribuiu como desempate para os métodos, porém contribuiu como critério de confirmação de que os melhores métodos possuem boa dispersão dos dados comparado com os

valores reais. A Tabela 5.7 apresenta os dados do resultado da aplicação dos 33 AMs para os dados transformados em múltiplos de 30 na base de dados do OE3.

Tabela 5.7 - AM aplicados a nova base de dados do OE 3

Método	Min	Max	MAE	MSE	R2	T - Treino (s)	T - Prev. (s)	IC (95%)
DTR	-1114,00	614,00	85,57	18686,20	0,99	<0,00	<0,00	(-23,76 ; 3,67)
ETC	-10740,00	6860,00	219,55	629212,96	0,98	1,81	0,06	(-35,14 ; 124,21)
DTC	-1734,00	1600,00	235,71	171413,52	0,99	0,23	<0,00	(-57,92 ; 25,32)
LDA	-8606,00	2640,00	273,89	555066,68	0,98	0,27	0,02	(-99,36 ; 50,45)
RFC	-1667,00	8673,00	288,00	730902,76	0,97	1,70	0,08	(78,72 ; 247,57)
GNB	-8046,00	1846,00	465,36	1088072,23	0,96	0,11	0,08	(-217,88 ; -9,26)
PR	-1426,23	1382,86	497,31	352118,64	0,99	0,05	0,02	(-47,15 ; 72,21)
GPR	-2071,52	5678,88	622,25	897757,59	0,97	68,30	0,09	(87,90 ; 275,01)
LR	-10060,00	8714,00	819,97	3486441,29	0,87	36,19	0,02	(-333,61 ; 40,91)
MLPR	-5421,44	5686,68	1589,58	4417892,37	0,84	0,28	<0,00	(-42,23 ; 379,30)
ABC	-3554,00	9534,00	1837,14	9117281,88	0,66	22,62	2,71	(715,35 ; 1288,45)
RFR	-4202,27	6897,73	1846,79	5870711,19	0,78	0,02	<0,00	(-295,71 ; 191,67)
MLPC	-11867,00	11733,00	1855,10	11345591,60	0,58	77,68	0,02	(-330,41 ; 347,29)
GTBR	-4525,54	5164,64	1959,24	6105273,66	0,77	0,03	<0,00	(33,73 ; 527,65)
KRR	-6315,53	5819,35	2282,12	7855876,55	0,71	10,62	<0,00	(-182,08 ; 381,48)
EN	-5911,44	4861,80	2282,63	7372661,98	0,73	<0,00	<0,00	(-21,50 ; 522,48)
ARDR	-6052,38	4686,46	2297,88	7356612,73	0,73	185,29	<0,00	(-18,84 ; 524,50)
BRR	-6049,77	4655,12	2298,47	7357996,06	0,73	<0,00	<0,00	(-17,08 ; 526,27)
RRCV	-6051,09	4653,14	2298,62	7358198,50	0,73	0,02	<0,00	(-17,05 ; 526,31)
RR	-6051,81	4652,08	2298,70	7358310,02	0,73	0,02	<0,00	(-17,03 ; 526,34)
LASSO	-6052,33	4651,16	2298,75	7358340,06	0,73	<0,00	<0,00	(-17,01 ; 526,36)
OLS	-6052,52	4651,01	2298,78	7358423,48	0,73	<0,00	<0,00	(-17,01 ; 526,36)
OMP	-5656,60	5170,37	2302,04	7479930,55	0,72	0,02	<0,00	(-45,36 ; 502,97)
SGD	-16420,00	8573,00	2462,57	14407342,65	0,46	22,96	0,02	(-1481,90 ; -752,03)
NCC	-12706,00	11807,00	2534,54	16107163,52	0,40	0,06	0,02	(-666,07 ; 139,67)
MNB	-10054,00	16434,00	2887,77	19755199,34	0,26	0,11	<0,00	(1112,55 ; 1951,99)
SVM	-9320,00	17500,00	3458,05	26143750,24	0,03	73,29	11,22	(2077,23 ; 2971,84)
ECOC	-5833,00	17514,00	3461,14	25567187,39	0,05	663,54	0,16	(2360,50 ; 3209,61)
BC	-13813,00	17527,00	3483,17	29526882,52	0,10	0,02	0,14	(1691,12 ; 2691,56)
NNC	-10753,00	17413,00	4055,89	32200424,86	0,20	<0,00	0,05	(2640,06 ; 3594,09)
BNB	-1834,00	18080,00	6611,43	68753153,65	0,15	0,11	0,02	(5954,65 ; 6996,61)
OVRC	-18514,00	8567,00	9578,37	118466672,04	0,03	40,20	0,05	(-9832,39;-8680,34)
OVOC	----------	----------	---------	----------	------	----------	----------	----------

Analisando os dados da Tabela 5.7, o método DTR foi o que melhor gerou as previsões das respostas para o OE3. Em comparação ao segundo melhor método, o ETC, o DTR foi 61,02% melhor no quesito MAE e obteve tempos de treinamento e de previsão inferiores a 0,01

segundos, enquanto o tempo de treinamento do ETC foi de mais de 1,8 segundos. A Figura 5.11 ilustra o *boxplot* para o erro dos 12 melhores métodos de acordo com a Tabela 5.7.

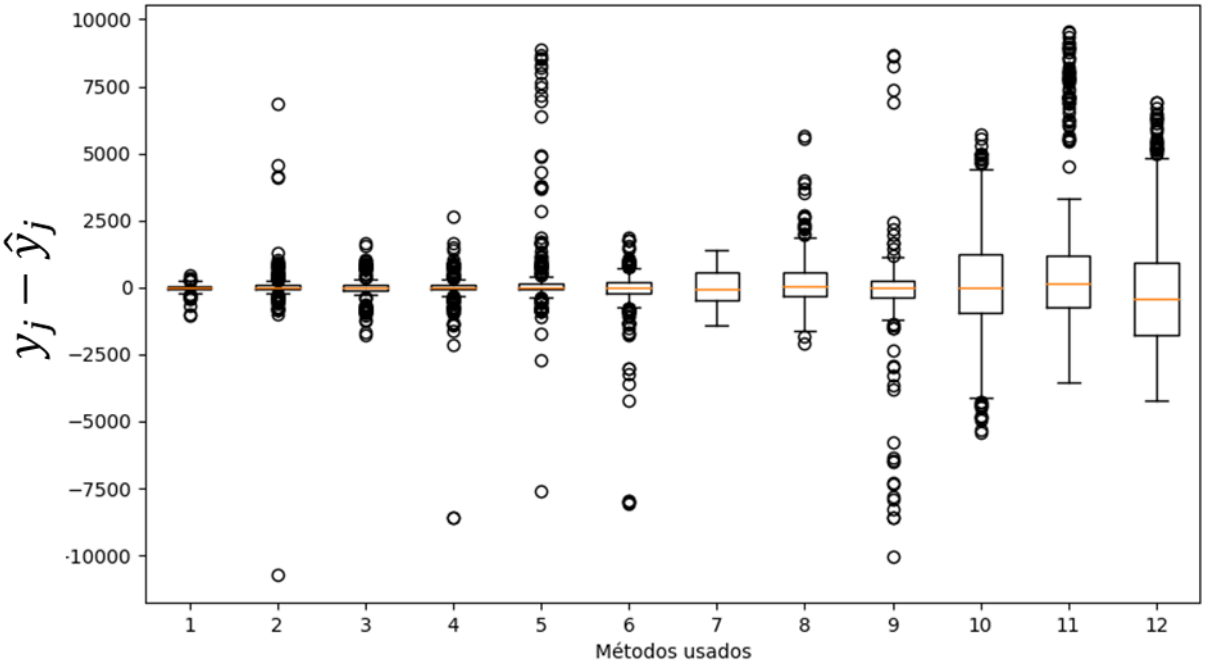

Figura 5.11 - *Boxplots* para o erro das 12 primeiras AM no OE3

Analisando a Figura 5.11, o *boxplot* para a AM DTR (método 1 no gráfico) é a que possui menor amplitude para o erro das previsões, fato este desejável para se gerar previsões com o menor erro possível, em comparação aos demais métodos. A Figura 5.12 ilustra a dispersão dos 12 melhores métodos de AM para o OE3.

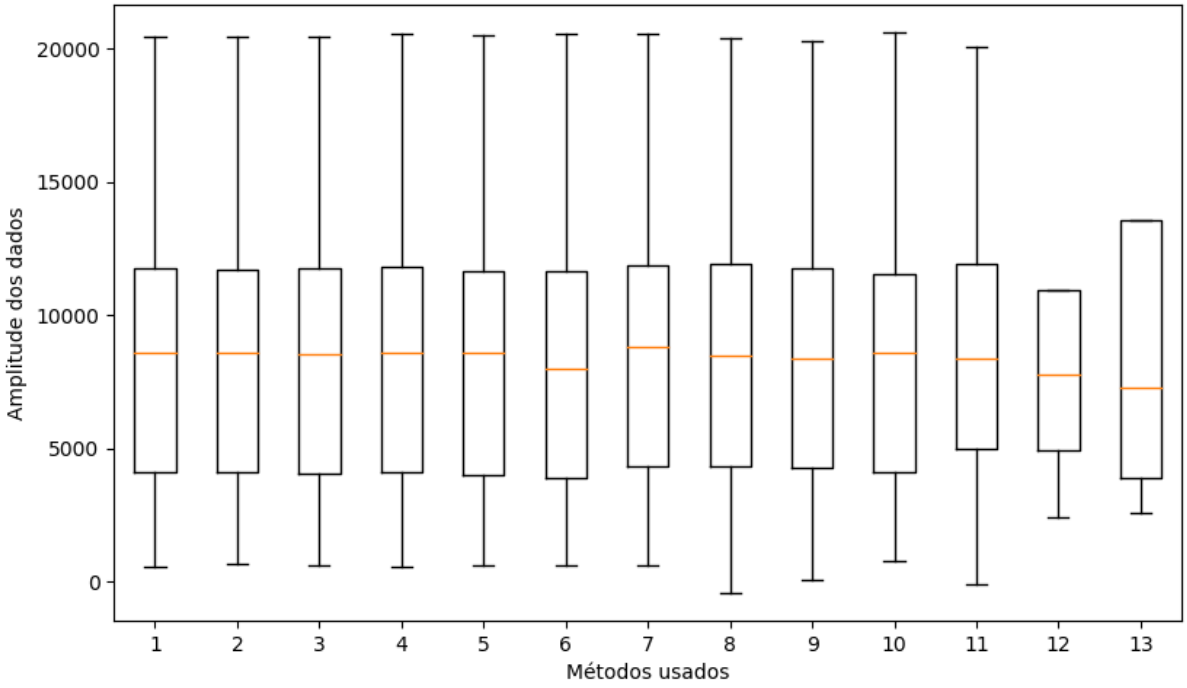

Figura 5.12 - Dispersão dos dados para os 12 primeiros métodos

Analisando a dispersão dos dados gerados para as previsões (Figura 5.12, métodos presentes nos *boxplots* de 2 a 13) com os dados reais (*boxplot* 1), os métodos que possuem melhores previsões, possuem a dispersão de seus dados próximos dos valores dos dados reais. Assim como nos OE1 e 2, a dispersão dos dados no gráfico da Figura 5.12 não define, mas ajuda a confirmar a qualidade das previsões geradas e do método de AM como possível metamodelo.

Desta forma, o DTC por ter sido a AM mais assertiva (menores MAE, MSE e intervalo de confiança para o erro) e com tempos de processamento menor dentre os demais, de acordo com a Tabela 5.7, e possuindo as menores dispersões de acordo com o *boxplot* da Figura 5.11, foi selecionado para compor o algoritmo de otimização do OE3.

Mesmo não garantindo uma total aderência, com erro de previsão ou classificação diferentes de zero por todos os métodos de AM apresentados, foi possível o teste e seleção de AMs capazes de serem utilizados como metamodelos da SED dos objetos de estudo. O fato de a AM ter gerado um erro baixo, se aproximando dos valores reais das FOs, corrobora com a possibilidade de se utilizar a AM como metamodelo para auxiliar na direção de quais soluções devem ser testadas com maior probabilidade de serem bons ótimos locais.

5.3 Geração e teste dos cenários das otimizações

A partir da seleção das metaheurísticas GA e GRASP e do teste das AMs com a escolha dos métodos GPR e PR para o OE1 e DTR para o OE2 e OE3, compete a essa parte da tese a definição dos cenários e teste de desempenho em comparação aos ótimos global e local, para os OE1, OE3 e OE2, respectivamente.

5.3.1 Teste e resultado de desempenho dos algoritmos no OE1

De acordo com o que foi proposto para os parâmetros de otimização e avaliação do problema no item 4.2.2, um total de 12 cenários foram gerados para teste do OE1, conforme a Tabela 5.8.

Tabela 5.8 - Cenários de otimização para o OE1

Algoritmo Genético (GA)	**Cenário**	**GRASP**	**Cenário**
1. GA sequencial	(GA-S)	2. GRASP sequencial	(GRASP -S)
3. GA paralelamente com população de tamanho 120	(GA-120)	4. GRASP paralelamente com população de tamanho 120	(GRASP - 120)
5. GA paralelamente com 1º AM e tamanho de previsão 120	(GA-GPR-120)	6. GRASP paralelamente com 1º AM e tamanho de previsão 120	(GRASP - GPR-120)
7. GA paralelamente com 1º AM e tamanho de previsão 1200	(GA-GPR-1200)	8. GRASP paralelamente com 1º AM e tamanho de previsão 1200	(GRASP - GPR-1200)
9. GA paralelamente com 2ª AM e tamanho de previsão 120	(GA-PR-120)	10. GRASP paralelamente com 2ª AM e tamanho de previsão 120	(GRASP - PR-120)
11. GA paralelamente com 2ª AM e tamanho de previsão 1200	(GA-PR-1200)	12. GRASP paralelamente com 2ª AM e tamanho de previsão 1200	(GRASP - PR-1200)

Um passo anterior a aplicação dos métodos de OvS e verificação dos resultados nos objetos de estudo é a definição de qual configuração computacional é mais adequada para a geração dos dados finais. Na Tabela 5.9, os resultados são apresentados para a aplicação das 12 instâncias para o OE1, nas duas máquinas especificadas na Seção 4.2.2.

Tabela 5.9 - Teste do desempenho dos algoritmos em diferentes máquinas

Cenários		**Tempo (segundos)**		**Ganho (%)**
		Processador: Xeon x2	**Processador: i7**	
Paralelo + AM (1200)	8. GRASP GPR 1200	2028	3061	33,75
	7. GA GPR 1200	2708	2761	1,92
	12. GRASP PR 1200	1725	1868	7,66
	11. GA PR 1200	2982	3061	2,58
Paralelo + AM (120)	10. GRASP GPR 120	2735	2708	-1,00
	9. GA PR 120	5144	5451	5,63
	6. GRASP PR 120	2565	2677	4,18
	5. GA GPR 120	3093	3143	1,59
Paralelo	4. GRASP 120	5336	5688	6,19
	3. GA 120	4950	4806	-3,00
Serial	2. GRASP Serial	17663	6604	-62,61
	1. GA Serial	42410	15541	-63,35

Analisando os dados da Tabela 5.9, é possível inferir que, em uma primeira análise, os resultados mostram que a Máquina 2 (i7) tem um melhor desempenho geral. Esse resultado foi uma consequência da melhor capacidade de processar instruções sequenciais com uma velocidade de *clock* maior para seu modelo do processador. Considerando apenas as instâncias de processamento paralelo, o resultado para o tempo médio é 5,95% menor na Máquina 1 (Xeon x2). Para os fins do restante deste estudo, os resultados presentes estão relacionados à Máquina 1 que possui um melhor desempenho em relação a um dos objetivos do trabalho em usar paralelismo em um mesmo ambiente de *hardware*.

As máquinas usadas para testes representam dois conjuntos diferentes disponíveis no mercado que são capazes de maximizar o processamento paralelo. Do ponto de vista da otimização, a máquina com mais núcleos e menor velocidade de *clock* tem um resultado 5,95% melhor. Do ponto de vista do investimento do projeto, a quantidade de tempo economizado nas instâncias de otimização utilizadas não justifica um investimento de US$ 15000,00 na máquina 1 contra os US$ 2000,00 na máquina 2. Para isso, é recomendável que, para trabalhos futuros, o teste do paralelismo em um ambiente distribuído com estações de trabalho de última geração é usado em vez de um único servidor novo.

A Figura 5.13 apresenta os resultados para a aplicação das instâncias da Tabela 5.8 relacionadas ao GA, respeitando o valor da melhor solução atual na iteração e o tempo para o final de cada iteração. A única instância que não é representada foi a serial, que demandou um tempo exponencial a ser avaliado e interferiu na escala da representação para os outros métodos.

A separação da representação gráfica dos resultados das instâncias de GA e GRASP ocorreu pela melhor visualização e análise dos mesmos.

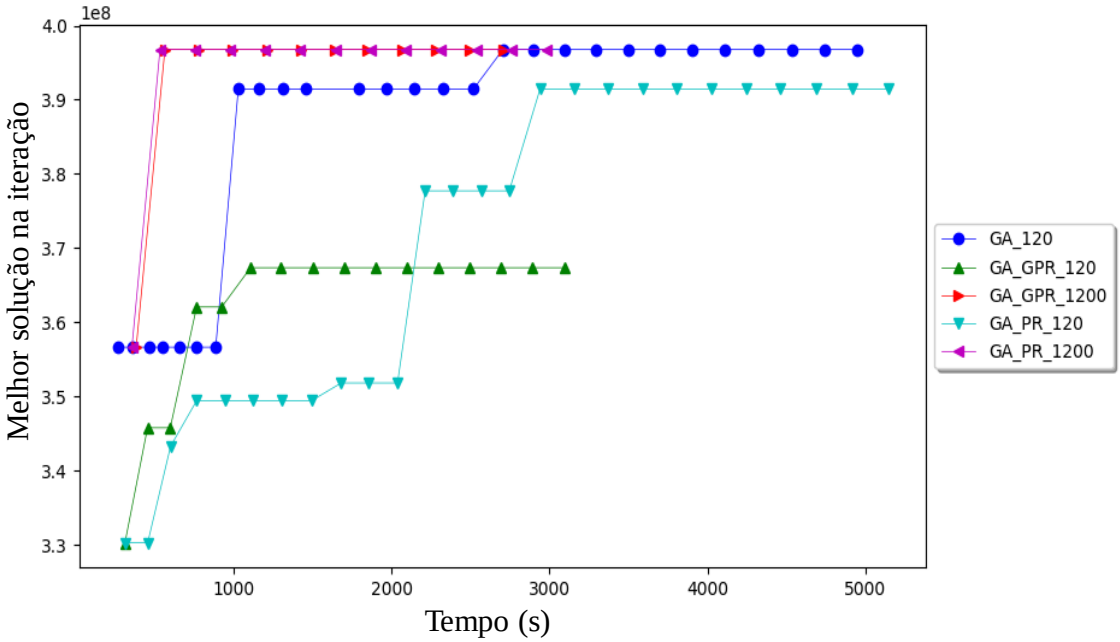

Figura 5.13 - Cenários com GA para OE1

De acordo com a Figura 5.13, o melhor cenário de otimização aplicando o GA é com a associação com a AM GPR e a avaliação de 1200 indivíduos em cada iteração (GA_GPR_1200). Os dados são apresentados através das setas vermelhas na Figura 5.13, com uma redução de tempo de 10,10% para o segundo melhor método de otimização de GA. O outro método que utilizou a AM PR com o maior nível de previsão (1200), encontrou uma solução 0,68% distante do ótimo global na segunda interação do algoritmo. A Figura 5.14 ilustra os dados para as instâncias que usaram a otimização GRASP e, pelas mesmas razões, não apresenta a instância para otimização serial.

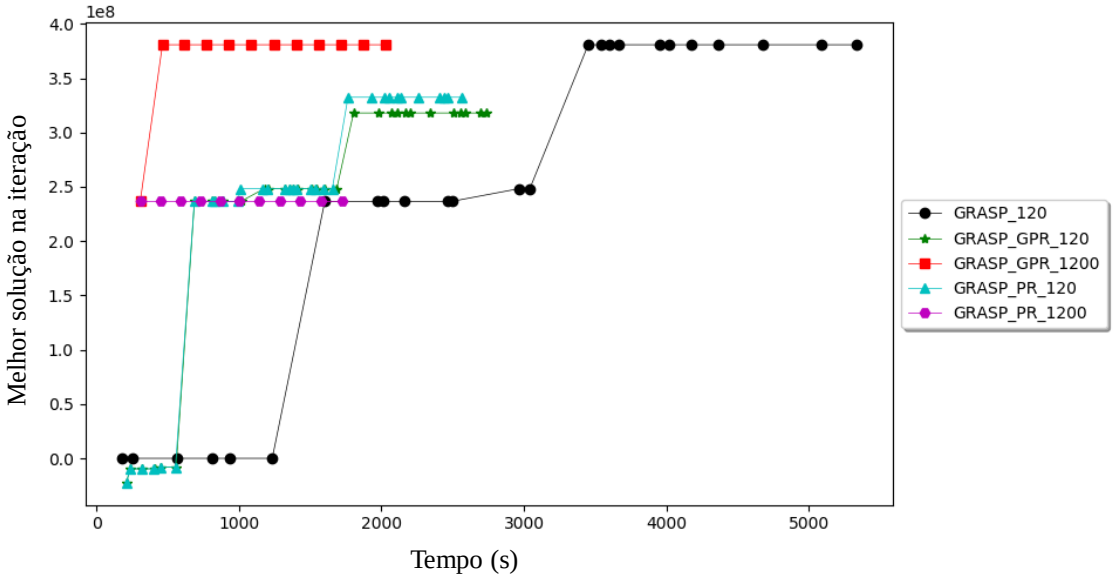

Figura 5.14 - Cenários com GRASP para OE1

Da mesma forma que ocorreu com o GA, no GRASP os melhores cenários de otimização estão relacionados ao uso de GPR / PR e 1200 previsões. Esse resultado corrobora com a mesma ideia de que o uso de metaheurística com um AM ajustado tem chance de encontrar boas soluções de maneira mais rápida, do que sem o uso de AM. A Tabela 5.10 apresenta os resultados gerais para as 12 instâncias de otimização avaliadas para uma ordenação e classificação final.

Tabela 5.10 - Resultado para os 12 cenários de otimização do OE1

Cenário	Melhor Solução		Tempo (s)		Pontuação final
	Valor	Pontua.	Valor	Pontua.	
8. GRASP GPR 1200	380811200	0,953	2028	0,851	0,811
7. GA GPR 1200	396770000	0,993	2708	0,637	0,633
12. GRASP PR 1200	236614800	0,592	1725	1,000	0,592
11. GA PR 1200	396770000	0,993	2982	0,578	0,574
10. GRASP PR 120	332468900	0,832	2565	0,673	0,560
5. GA GPR 120	367363300	0,920	3093	0,558	0,513
6. GRASP GPR 120	317995100	0,796	2735	0,631	0,502
3. GA 120	396770000	0,993	4950	0,348	0,346
9. GA PR 120	391436700	0,980	5144	0,335	0,328
4. GRASP 120	380811200	0,953	5336	0,323	0,308
1. GA Serial	396770000	0,993	42410	0,041	0,041
2. GRASP Serial	94200900	0,236	17663	0,098	0,023
Referência: Ótimo global	399473800	1,000	1900800	-----	-----

Considerando que o tempo de processamento e o valor da função objetivo possuem o mesmo peso e foram atribuídos com valores de 100%, as pontuações finais definem que o cenário GRASP GPR 1200 gerou a melhor relação de tempo e solução gerada, seguida pelos três cenários que usaram AM e 1200 previsões para cada interação. Comparado com o ótimo

global, a melhor otimização gerou uma solução 4,68% menor que a global, com um tempo próximo ao mínimo testado. Em uma comparação direta do melhor exemplo que utilizou apenas uma metaheurística em série (série GRASP), metaheurística com paralelismo (GA 120), o primeiro nível de previsão em AM (GRASP PR) e o segundo nível de previsão (GRASP PR 1200) representam uma "escada" de ganhos comparado ao processamento serial. Os ganhos são de 72%, 85,5% e 88,5% para o tempo e, em comparação com o pior e o melhor cenário, uma melhoria de tempo de 95,2% foi alcançada. A Figura 5.15 apresenta o gráfico de comparação dos resultados para a FO na escala da direita em comparação com o ótimo global, e do tempo na escala da esquerda.

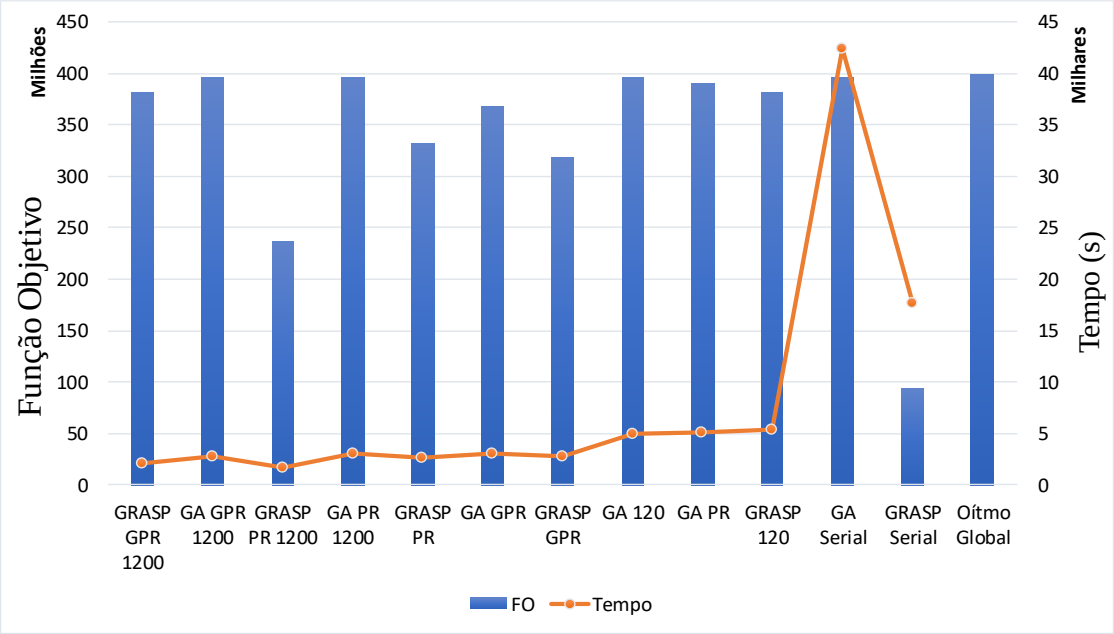

Figura 5.15 - Gráfico de barras e linha para OvS do OE1

Fazendo um comparativo com os dados presentes na Tabela 5.10 e Figura 5.14, os ganhos médios em relação ao tempo de processamento foram de 88,7% relativo aos cenários que utilizaram processamento serial *versus* o paralelo (cenários 1-2 vs. 3-12), de 15,1% comparado a média dos cenários que utilizaram paralelismo com e sem AM (cenários 3 e 4 vs. 5 a 12) e de 90,4% em comparação dos cenários seriais para com os cenários que utilizaram paralelismo e AM (cenários 1 e 2 vs. 4 a 12).

Não é o foco do presente trabalho a análise do desempenho da utilização isolada de otimização por metaheurísticas, mas cabe ressaltar que dentre os melhores resultados, o GA com AM e paralelismo em média teve resultados para a FO melhores que o GRASP com AM e paralelismo, porém com tempos mais elevados. Parte desse resultado pode ser explicado pelo

fato do GA possuir duas funções que utilizam de aleatoriedade (seleção de pais e mutação) contra apenas uma para o GRASP (seleção de pais), assim um algoritmo mais complexo (maior quantidade de linhas de código) pode ser mais eficiente em caminhar no espaço solução para encontrar boas soluções, porém um algoritmo menos complexo pode ser mais rápido em encontrar respostas próximas de um nível desejado de solução.

5.3.2 Teste e resultado de desempenho dos algoritmos no OE2

Assim como realizado para o OE1, de acordo com o que foi proposto para os parâmetros de otimização e avaliação do problema no item 4.2.2, um total de 6 cenários foram gerados para teste do OE2, conforme a Tabela 5.11.

Tabela 5.11 - Cenários de otimização para o OE2

Método	Cenário
1. GRASP sequencial	(GRASP - Serial)
2. GRASP paralelo com população de tamanho 120	(GRASP - Paralelo)
3. GRASP paralelo com DTR e tamanho de previsão 120	(GRASP - DTR_x1)
4. GRASP paralelo com DTR e tamanho de previsão 1200	(GRASP - DTR_x10)
5. GRASP paralelo com DTR e tamanho de previsão 12000	(GRASP - DTR_x100)
6. GRASP paralelo com DTR e tamanho de previsão 120000	(GRASP - DTR_x1000)

De acordo com a Tabela 5.11, quatro níveis do tamanho da previsão do vetor de metamodelagem por AM foram definidos, sendo x1, x10, x100 e x1000 o tamanho da população inicial. Estes níveis do tamanho da previsão foram selecionados para se testar o efeito na qualidade da resposta quanto ao tempo e a distância para a solução de referência com o aumento do número de previsões por iteração.

A Figura 5.16 apresenta os resultados para a aplicação dos cenários apresentados na Tabela 5.9, ilustrando o valor da melhor solução na iteração corrente e o tempo para o final de cada iteração. Os dois únicos cenários que não foram representados na íntegra foram o serial e o DTR x1000 que demandaram um tempo exponencial a ser avaliado (Tabela 5.12) e interferiram na escala da representação para os demais métodos, que apresentaram apenas os três valores iniciais.

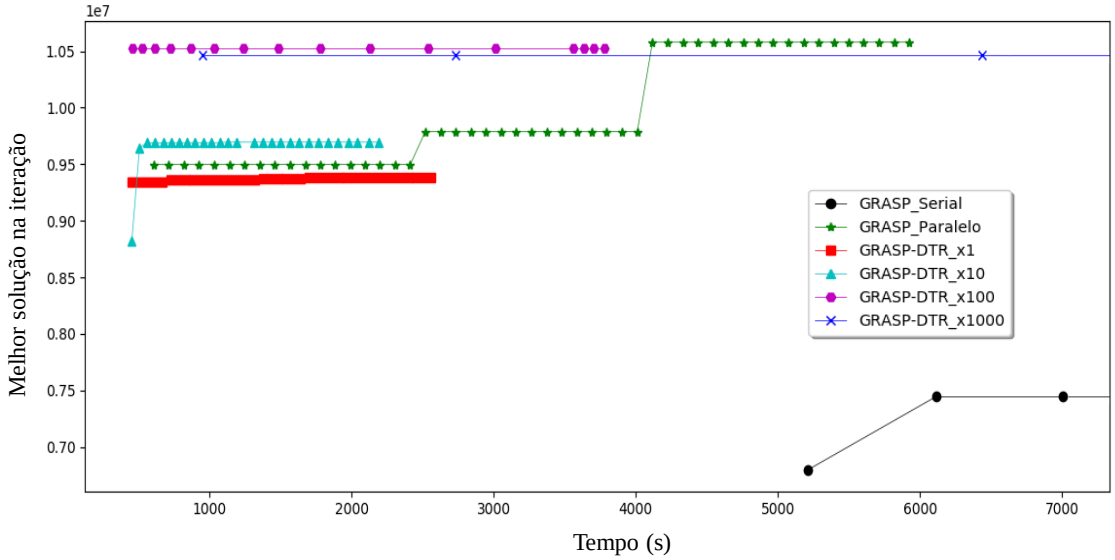

Figura 5.16 - Cenários e resultado de OvS para OE2

De acordo com a Figura 5.16, os cenários de otimização que geraram as melhores soluções foram os que utilizaram de paralelismo, ou aqueles que tiveram AM com x100 e x1.000 iterações antes da avaliação da simulação. Isso pode levar a uma conclusão antecipada sem a análise do tempo relacionado. Para uma classificação final, os métodos foram pontuados para os melhores valores de função objetivo encontrados e o tempo gasto com o mesmo peso para ambas as variáveis. Os resultados são apresentados na Tabela 5.12.

Tabela 5.12 - Resultado para os seis cenários de otimização do OE2

Cenário	**Solução final**			**Tempo (s)**		**Rank Final**
	Solução (lot;seg;rev)	**Função objetivo**	**Rank**	**Valor**	**Rank**	
4: GRASP DTR x10	1927; 85; 4800	9695122	0,899	2193	1,000	0,899
3: GRASP DTR x1	1598; 82; 2700	9385339	0,871	2559	0,857	0,746
5: GRASP DTR x100	1568; 07; 3000	10528161	0,977	3778	0,580	0,567
2: GRASP Paralelo	1475; 69; 5400	10578676	0,981	5925	0,370	0,363
6: GRASP DTR x1000	1384; 12; 6000	10465319	0,971	30656	0,072	0,069
1: GRASP Serial	1455; 57; 7500	10162864	0,943	43399	0,051	0,048
Referência: Busca randômica	1570; 12; 2100	10778862	1,000	964800	-----	-----

A Tabela 5.12 apresenta valores finais e a classificação dos cenários testados, definindo o uso do GRASP, juntamente com 10 replicações por solução, e avaliação utilizando o DTR antes da chamada da simulação paralela, para gerar o melhor conjunto de parâmetros de otimização avaliando a FO e o tempo. Como resultado, o uso de x10 iterações para gerar vizinhos a serem avaliados usando as previsões da AM DTR é 17%, 37% e 92,3% melhor que

o uso de x1, x100 e x1.000 iterações, respectivamente. Esse efeito pode ser explicado pelo ponto de equilíbrio gerado pelo *trade-off* entre o tempo e o valor da FO.

Por essa razão, um pequeno número de previsões, iteração = x1, gerou apenas vizinhos para as soluções atuais que possuem valores de FO que são muito semelhantes entre si. Este fato tem impacto direto no tempo computacional, pois gerou um maior número de predições com valores semelhantes que exigiram uma avaliação paralela ao final da iteração. De maneira oposta, o uso de um grande número de iterações (x1000) antes da avaliação da simulação, demandou uma quantidade maior de tempo (pelo menos 2700 segundos) em cada iteração.

Avaliando a qualidade da resposta dos cenários de forma gerencial, o cenário "GRASP DTR x100" configurou uma boa solução, pois são decrementados o tamanho do lote econômico e o nível de estoque de segurança, que configura um aumento de 37,5% no giro do estoque durante o período de revisão. Para encontrar esta solução, foi necessário mais 42% de tempo, o que representa apenas 26,4 minutos a mais do que a melhor solução anterior, com um aumento de 7,9% no valor da FO.

Quanto ao uso do paralelismo, o ganho foi de 86,3% e 86,8% relacionado ao tempo e a pontuação final da otimização seriada (cenário 1 vs. 2). Em comparação, os três melhores resultados de solução foram em média 50,7% melhores que os paralelos (cenários 3 a 5 vs. 2). Portanto, o uso de ambos os métodos gerou uma melhoria média de 93,5% na pontuação final, comparando uma abordagem mais tradicional de usar apenas a metaheurística GRASP de forma serial para o estudo de objeto proposto (cenários 3 a 5 vs. 1). A Figura 5.17 ilustra com gráficos de barras e linha os resultados da Tabela 5.12.

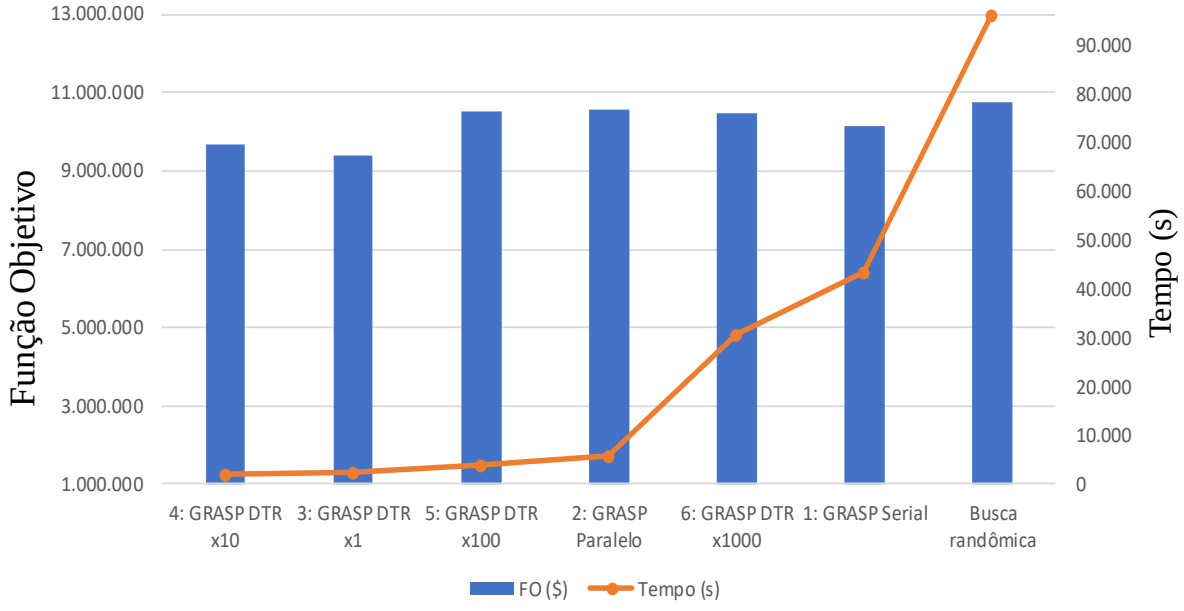

Figura 5.17 - Gráfico de barras e linha para OvS do OE2

Os gráficos de barras e linha presentes na Figura 5.17 reforçam a ideia do ponto de equilíbrio gerado entre o valor da FO e o tempo necessário para se encontrar a resposta, de maneira tal que a consideração de mais tempo de processamento, uma melhor solução pode ser gerada, passando do cenário 4 para o cenário 5 de combinação de OvS, dessa forma, há uma sensibilidade na variável "tamanho da previsão por metamodelagem usando AM".

O tempo da "solução randômica" foi reduzida para 10% de seu valor (96800 s) em sua representação gráfica, para que com essa nova escala fosse possível a apresentação da diferença entre os cenários de otimização. A partir da análise do tempo, o fator "tamanho da previsão" também é um fator importante para a avaliação do binômio FO e tempo, no qual o cenário 6 se torna tão caro computacionalmente, quanto o cenário 1.

5.3.3 Teste e resultado de desempenho dos algoritmos no OE3

De maneira análoga ao realizado para o OE2, de acordo com o que foi proposto para os parâmetros de otimização e avaliação do problema no item 4.2.2, seis cenários foram gerados para teste do OE3, conforme a Tabela 5.13.

Tabela 5.13 - Cenários de otimização para o OE3

Método	Cenário
1. GRASP sequencial	(GRASP - Serial)
2. GRASP paralelo com população de tamanho 120	(GRASP - Paralelo)
3. GRASP paralelo com DTR e tamanho de previsão 120	(GRASP - DTR_x1)
4. GRASP paralelo com DTR e tamanho de previsão 1200	(GRASP - DTR_x10)
5. GRASP paralelo com DTR e tamanho de previsão 12000	(GRASP - DTR_x100)
6. GRASP paralelo com DTR e tamanho de previsão 120000	(GRASP - DTR_x1000)

Os cenários de OvS para o OE3 de acordo com a Tabela 5.13 foram propostos para se avaliar de forma separada o resultado dos métodos de otimização e comparar o efeito dos mesmos no que tange ao resultado obtido e ao tempo necessário para se chegar na resposta com no máximo 12 iterações sem melhora para o ótimo obtido até o momento de cada iteração. Assim, como para os OE1 e OE2, foram testados 4 níveis para o tamanho da quantidade de previsões por iteração, de forma tal a testar e aproveitar a melhor metamodelagem por AM.

A Figura 5.18 apresenta os resultados para a aplicação dos cenários apresentados na Tabela 5.13, relaciona o valor da melhor solução na iteração corrente e o tempo para o final de cada iteração. Os dois únicos cenários que não foram representados na íntegra foram o GRASP

serial e o GRASP DTR x1000 que demandaram um tempo exponencial a ser avaliado (Tabela 5.14) e interferiram na escala da representação para os demais métodos.

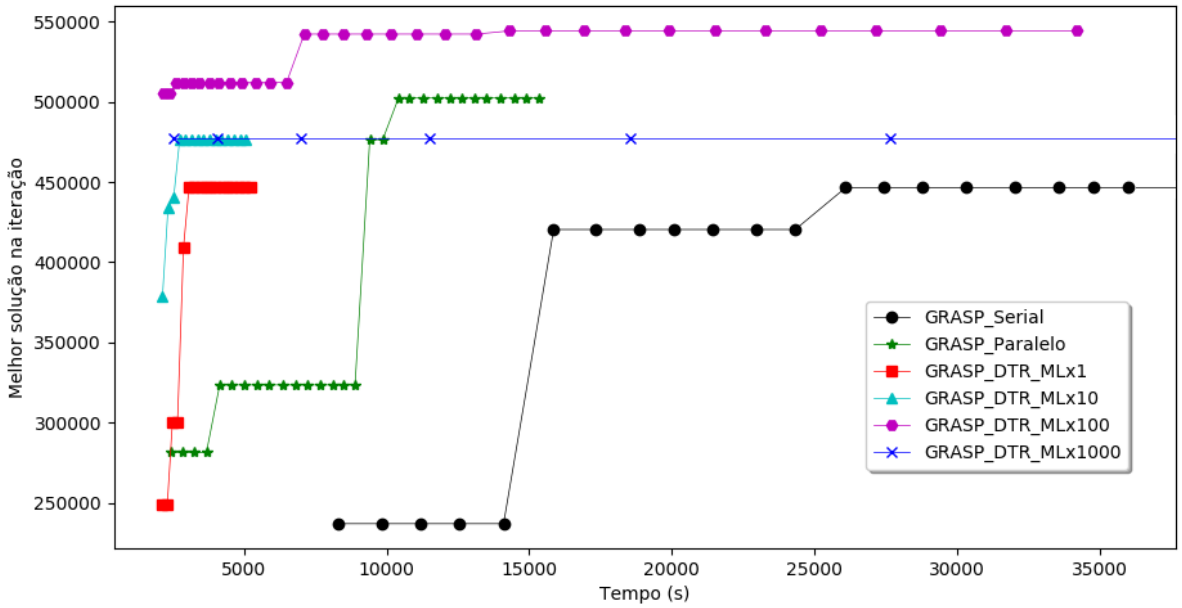

Figura 5.18 - Cenários e resultado de OvS para OE3

De acordo com a Figura 5.18, os cenários de otimização que geraram as melhores soluções foram: com a utilização de paralelismo ou da AM DTR com x100 e x1.000 iterações antes da avaliação da simulação. Essa avaliação preliminar e visual pode levar a uma conclusão errônea sem a análise do tempo relacionado. Para uma classificação final, os métodos foram pontuados para os melhores valores de função objetivo encontrados e o tempo gasto com o mesmo peso para ambas as variáveis. Os resultados são apresentados na Tabela 5.14, tanto para os cenários propostos na Tabela 5.13, quanto para a resposta ótima proposta em Law e Kelton (2007).

Tabela 5.14 - Resultado para os seis cenários de otimização do OE3

Cenário	Solução final ($)			Tempo (s)		Rank Final
	Solução $(M_1; B_1; M_2; B_2; M_3; B_3; M_4;)$	Função objetivo	Rank	Valor	Rank	
4: GRASP DTR x10	3;06;3;01;3;01;3	476.280	0,770	5495	1,000	0,770
3: GRASP DTR x1	2;01;3;01;2;01;2	446.500	0,722	5607	0,980	0,707
2: GRASP Paralelo	3;06;3;05;1;01;2	502.240	0,812	16.187	0,339	0,276
5: GRASP DTR x100	3;02;3;01;1;01;2	544.060	0,879	39.356	0,140	0,123
1: GRASP Serial	3;03;3;01;1;01;3	542.160	0,876	70.704	0,078	0,068
6: GRASP DTR x1000	3;06;3;01;1;06;3	476.900	0,771	194.949	0,028	0,022
Law e Kelton	3;07;3;09;2;04;2	591.600	0,956	-----	-----	-----
Referência: Ótimo global	3;07;3;x;1;9-x;2 para $1 \leq x \leq 8$	618.800	1,000	777.600	-----	-----

Analisando a resposta gerada pelos autores Law e Kelton com o auxílio do *software* Witness®, a configuração no ambiente utilizado pelos mesmos retornou uma FO igual a $587290. Com a mesma configuração (demonstrada na Tabela 5.14), o ambiente de simulação Simpy retornou uma FO igual a $591600, cerca de 0,73% maior. Considerando a diferenças entre os *softwares* utilizados, e que no referido livro os autores deixam claro que esta é uma boa resposta, porém sem definir se o mesmo é o ótimo global do mesmo, sem fazer referências de quanto tempo computacional empregado ou qual o *hardware* utilizado para o mesmo, esta resposta serve apenas de comparação intermediária, sem efeito de comparação final.

De maneira análoga ao OE2, o uso da OvS GRASP com paralelismo e metamodelagem pela AM DTR e tamanho de previsão x10, foi o método de otimização que gerou a melhor resposta para o OE3, considerando a relação entre a resposta e o tempo necessário para se chegar no mesmo. Considerando apenas o valor da FO, o cenário 4 foi o quinto colocado (comparando também com a resposta de Law e Kelton), porém seu tempo de processamento foi inferior aos demais, gerando um peso que superou os demais na classificação final.

Do ponto de vista prático, analisando o tempo de processamento, as duas melhores OvSs (cenários 4 e 3) consumiram cerca de 1,5 horas de processamento computacional, gerando respostas em média 74,6% do ótimo global. Com o acréscimo de 9,4 horas ao tempo de processamento (total de 10,9 horas), o cenário 5 gera uma resposta cerca de 87,9% próximo do ótimo global. Em um projeto de Engenharia de Produção para a avaliação de novos postos de trabalho ou melhoria dos já existentes, o acréscimo de 9,4 horas não fornece um impedimento *a priori*, com a vantagem de proporcionar uma configuração que gera um lucro adicional mensal de $67780, ou 14,23% maior que o gerado pelo cenário 4. A Figura 5.19 ilustra com gráficos de barras e linhas os resultados da Tabela 5.14.

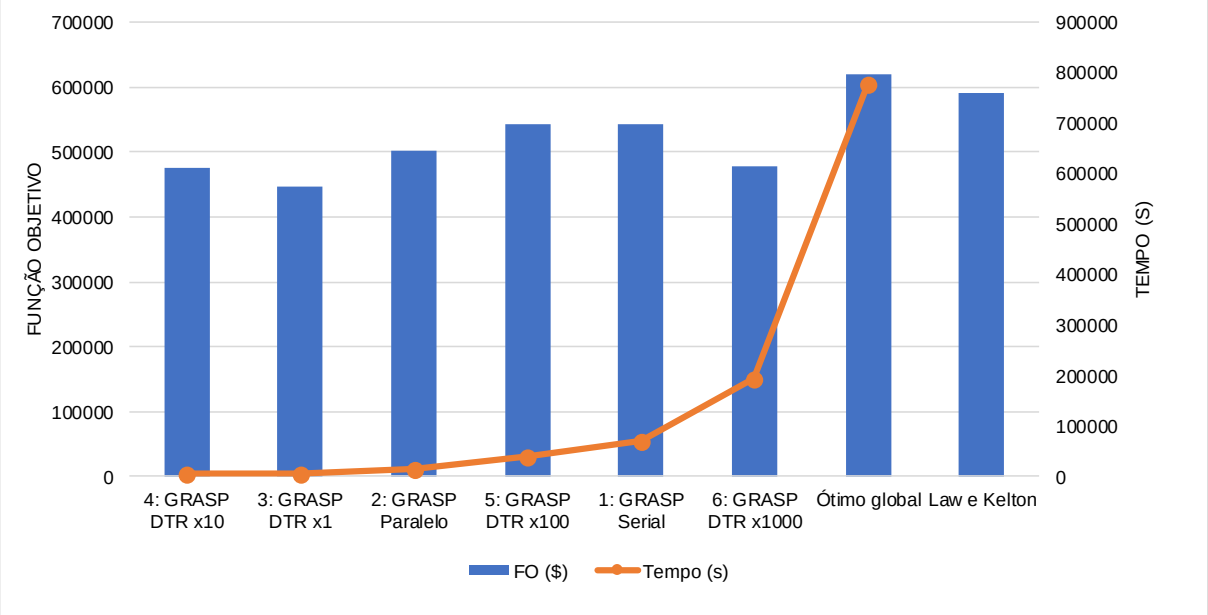

Figura 5. 19 - Gráfico de barras e linha para OvS do OE3

Os gráficos de barras e linha presentes na Figura 5.19 reforçam a ideia da análise de um possível "ponto de equilíbrio" gerado entre o valor da FO e o tempo necessário para se encontrar a resposta. Considerando a utilização de mais tempo de processamento, uma solução melhor (maior FO) pode ser gerada, passando do cenário 4 como ótimo para o cenário 5 de combinação de OvS, dessa forma há uma sensibilidade na variável "tamanho da previsão por metamodelagem usando AM" (assim como no OE2), de acordo com o tempo disponível para os testes do OvS.

6. CONCLUSÃO

No presente Capítulo é apresentado a avaliação do atendimento dos objetivos definidos no Capítulo 1, as considerações finais de acordo com as percepções do autor durante o desenvolvimento do trabalho e a proposta de futuros trabalhos que poderão surgir a partir dos resultados da atual pesquisa.

6.1 Verificação dos objetivos específicos

No Capítulo 1 foram definidos seis objetivos específicos para desenvolver o objetivo geral.

1. Aplicar técnica de Revisão Sistemática da Literatura para justificar a escolha dos métodos de otimização e quais os tipos de objeto de estudo seriam utilizados.

Este objetivo foi desenvolvido no Capítulo 2 com o intuito de se levantar o estado da arte no que se refere às técnicas de OvS e aos problemas e contextos em que os mesmos eram empregados. Com esse levantamento foi possível identificar quais técnicas eram mais empregadas e tinham a possibilidade de serem integradas em um método de otimização inovador e quais tipos de problemas eram mais recorrentes na Engenharia de Produção para serem selecionadas para teste desse método de OvS.

2. Propor método de otimização que englobe as principais técnicas de otimização identificadas na revisão sistemática, de maneira tal que seja possível a interação desses conceitos em um mesmo ambiente.

A partir da revisão sistemática foi possível a identificação das técnicas de metaheurística utilizando GA e GRASP, metamodelagem por AMs (utilizando as AMs com melhores resultados), com paralelismo via CPU. Essa escolha de métodos se deve pelo fato de aproveitamento das melhores características de cada método, considerando: uma busca mais rápida e "inteligente" por caminhar no espaço solução, utilizando de metaheurística; usar um modelo substituto por AM em vez de sempre rodar toda a simulação para se prever o valor da FO e utilizando de aumento de processamento computacional, para paralelizar as chamadas de simulação quando necessárias, com processamento em paralelo.

3. Propor e programar ambiente de otimização de maneira que seja possível a integração do método de otimização com os objetos de estudo.

Por serem técnicas distintas, cada uma necessita de especificidades como ambiente de programação flexível para o desenvolvimento dos códigos das metaheurísticas e fácil geração, armazenamento e tratamento de dados para a aplicação das técnicas de AM, além da integração com um sistema de modelagem computacional para a SED. Todas estas características foram possíveis de serem desenvolvidas com a linguagem de programação Python, e seus respectivos pacotes computacionais, assim como apresentado no Capítulo 4.

4. Gerar objetos de estudo a partir da revisão da literatura de maneira que representem problemas da Engenharia de Produção.

De acordo com a RSL realizada no Capítulo 2, foram desenvolvidos no Capítulo 4, três objetos de estudo com o intuito atender três fatores importantes: 1) representar 40,0% dos problemas de Engenharia de Produção, no que tange o "Chão-de-fábrica" e definição de "Lote Econômico" com os problemas de definição de recursos para chão-de-fábrica (OE1), definição de parâmetros para lote-econômico (OE2) e determinação da quantidade de máquinas e tamanho de *buffer* (OE3); 2) gerar ponto de comparação para o primeiro e terceiro objetos de estudo foi possível gerar o ótimo global e o segundo apenas um ótimo local de comparação; 3) definir três regiões uma para cada objeto de estudo respectivamente, onde as soluções ótimas se encontram, uma perto dos limites dos valores das variáveis (OE1) e as outras duas (OE2 e OE3) na região central do espaço solução.

5. Aplicar o método de otimização proposto dentro do ambiente computacional desenvolvido.

Após a definição do ambiente de programação, escrita dos códigos dos algoritmos de otimização e dos objetos de estudo, foi possível a integração dos mesmos, de maneira tal que os dados eram armazenados em arquivos CSV. A partir daí, informações e estatísticas foram extraídas para análise dos resultados.

6. Avaliar os resultados obtidos com a aplicação dos métodos em objetos de estudo relacionados à problemas de Engenharia de Produção, com a confirmação da redução do tempo computacional necessário para se encontrar "boas soluções".

A partir do método proposto no Capítulo 4 e os resultados apresentados no Capítulo 5, foi possível verificar o resultado positivo para a integração dos métodos de otimização selecionados. De acordo com os resultados apresentados na Seção 5, tem-se que em média para os objetos de estudo, as configurações de soluções serial, paralela e paralela com

metamodelagem por AM geraram respostas 76,2%, 93,5% e 87,5% respectivamente, em comparação com as soluções de referência considerada ótima global (OE1 e OE3) ou local (OE2), comprovando o potencial da metodologia proposta nesses três objetos de estudo.

Comparando o ganho de tempo computacional, da programação serial para a paralela e da paralela para a paralela com metamodelagem por AM, a média do ganho do tempo das melhores soluções foram de 82,1% e 63,2%, respectivamente, com uma economia de 93,5% para o tempo comparando processamento sequencial e a paralela com metamodelagem.

6.2 Considerações finais

Esta tese pode ser comparada aos esforços de otimização combinatória para a construção de novas hiper-heurísticas que possam aproveitar as técnicas de metaheurística, AM e o paralelismo. Esses esforços se juntam a uma das ideias da indústria 4.0 para o desenvolvimento de métodos para lidar com *big data*, identificar padrões ocultos e transformar informações úteis em ferramentas de tomada de decisão no ambiente industrial. Não era foco desta pesquisa mostrar a integração da OvS com a indústria 4.0, mas a inferência dessa relação serve para guiar trabalhos futuros, no que tange a utilização de ferramentas computacionais para gerar sistemas supervisórios que possam analizar múltiplos cenários e sugerir aqueles que melhor possarm gearar resultados para a indústria.

Foi possível a integração de diferentes métodos de otimização no que tange as principais práticas e tendências encontradas na literatura técnica especializada, no que se refere a otimização da simulação a eventos discretos aplicados a três objetos de estudo relacionados à problemas recorrentes da Engenharia de Produção. Tal configuração de integração de métodos diferentes foi capaz de gerar resultados satisfatórios para os problemas propostos, com a expressiva redução do tempo computacional e obtenção de respostas próximas de um ótimo de referência, com probabilidade de serem replicados para outros problemas a partir do método proposto e da utilização das ferrametas descritas.

Dentre os desafios enfrentados para a elaboração do presente trabalho, cabe ressaltar um problema que demandou muita pesquisa e tempo para ser solucionado. Projetos parecidos, ou iguais a esse, demandam a utilização de algum software de SED para que, de alguma forma, este gere um modelo computacional que será chamado recursivamente para geração de dados. A chamada recursiva do simulador de SED, de forma serial e paralela, foi um problema encontrado em softwares comerciais e solucionado com a programação *open source*. Dessa

maneira, recomenda-se para acadêmicos ou pessoas práticas da área, que ao iniciar um projeto de tal natureza, pesquise e escolha um *software* de SED que permita a integração de recursividade com o algoritmo de otimização desejado.

Para a elaboração e teste do ambiente de otimização foi necessário o aprendizado de linguagem de programação. Nesse quesito várias linguagens foram estudadas e testadas a exemplo de Java, C++ e C#, porêm a linguagem Python se mostrou aplicável em muitos casos científicos e também para a presente tese. Com a mesma, foi possível juntar de forma confiável vários conceitos que naturalmente são aplicadas de forma separada, e com a versatilidade dessa linguagem, foi possível criar o ambiente desejado, integrando SED, paralelismo, aprendizagem de máquina e metaheurística.

O fato de estar presente em um grupo de pesquisa multidisciplinar com foco na SED, foi importante para o amadurecimento dos conceitos aqui envolvidos, a partir da troca de ideias e discuções a respeito da metadologia aqui desenvolvida e empregada na presente tese. Além do fato da afirmação e fortalecimento da rede de pesquisadores na área de conhecimento da OvS, com o intuito de perdurar o mesmo para trabalhos além do tempo de doutoramento (ou mestrado) dos mesmos.

A participação e apresentação em congressos e eventos nacionais permitiu a compreensão do cenário nacional de pesquisa nessa área, com a comprovação de que estudos correlatos vem sendo desenvolvidos por outros centros de pesquisa, corroborando com a ideia de desenvolviemnto e continuidade da presente pesquisa em estudos atuais e futuros. Esse intercâmbio de ideias e trabalhos reforça a atualidade do tema e que o mesmo ainda possui apelo para ser estudado a longo prazo.

6.3 Sugestões de trabalhos futuros

A partir da elaboração do presente trabalho foram identificadas as seguintes oportunidades para a realização de trabalhos futuros:

- De acordo com os dados presentes na Seção 2.6, não foi encontrado trabalho de modelagem, otimização e principalmente a implantação da solução ótima. Pode assim ser um projeto de implantação e comparação via ferramentas estatísticas dos resultados uma proposta de trabalho;
- De acordo com os dados da Seção 2.6.1, é fator de ineditismo a aplicação da OvS em outras áreas, a exemplo do setor primário pouco explorado por tais métodos;

- Também por inédito do presente trabalho, é considerado a junção das ferramentas de otimização e dos pacotes computacionais utilizados. Dessa forma, os mesmos podem ser testados com diferentes configurações;
- O método de análise e seleção da AM mais apta para ser inserida na metaheurística foi feita de forma separada do algoritmo principal de otimização. Em trabalhos futuros é interessante testar a seleção da AM, utilizando da população inicial gerada no início da metaheurística populacional, juntando tanto a seleção da AM com a OvS em um mesmo programa, testando para tal a relação entre o tamanho da população inicial com a taxa de assertividade da AM selecionada;
- Foi testado apenas uma medida de tratamento dos dados para a redução da variabilidade (divisão por 300000 ou 30) e construção das bases de dados para a aplicação das AMs. Em trabalhos futuros poderão ser testados outros índices de tratamento de dados, a fim de se otimizar os valores de referência para MAE, R^2 e IC das AM, a um nível aceitável de diferença entre as respostas, considerando a opinião de um agente decisor. Devido ao grande número de métodos de AM testados e utilizados, o teste de diferentes parâmetros para as AMs poderá ser desenvolvido em trabalhos futuros, assim como outros métodos e parâmetros de métodos metaheurísticos;
- Um novo trabalho pode ser gerado com o teste em outras configurações de hardware, expandindo a discussão para a utilização de paralelismo de SED em máquinas distribuídas ou com auxílio de processor gráfico (GPU);
- Integração da OvS com sistemas *online* de produção para a otimização de parâmetros industriais em tempo real é uma temática que surgiu como possível trabalho com demanda imediata de aplicação, assim como a sua importância para a indústria 4.0;
- Integração da metodologia proposta de otimização para ser implementada em conjunto com *softwares* comerciais de SED. Testar para modelagem por Agentes e Monte Carlo;
- Teste em outros problemas relacionados à Engenharia de Produção ou áreas do conhecimento que possuam características parecidas com as apresentadas na tese.

REFERÊNCIAS BIBLIOGRÁFICAS

ADEGOKE, A.; TOGO, H.; TRAORE, M. K. A unifying framework for specifying DEVS parallel and distributed simulation architectures. **Simulation-Transactions of the Society For Modeling and Simulation International**, v. 89, n. 11, p. 1293–1309, 2013.

AL-AOMAR, R.; AL-OKAILY, A. A GA-based parameter design for single machine turning process with high-volume production. **Computers and Industrial Engineering**, v. 50, n. 3, p. 317–337, 2006.

ALBA, E. (ED.). **PARALLEL METAHEURISTICS A New Class of Algorithm**. [s.l.] John Wiley & Sons, Inc., 2005.

ALBA, E.; LUQUE, G.; NESMACHNOW, S. Parallel metaheuristics: Recent advances and new trends. **International Transactions in Operational Research**, v. 20, n. 1, p. 1–48, 2013.

ALRABGHI, A.; TIWARI, A. A review of simulation-based optimisation in maintenance operations. **Proceedings - UKSim 15th International Conference on Computer Modelling and Simulation, UKSim 2013**, p. 353–357, 2013.

ALRABGHI, A.; TIWARI, A. State of the art in simulation-based optimisation for maintenance systems. **Computers and Industrial Engineering**, v. 82, p. 167–182, 2015.

AMOUZGAR, K.; BANDARU, S.; NG, A. H. C. Radial basis functions with a priori bias as surrogate models:A comparative study. **Engineering Applications of Artificial Intelligence**, v. 71, n. November 2017, p. 28–44, maio 2018.

AVCI, M. G.; SELIM, H. A multi-objective simulation-based optimization approach for inventory replenishment problem with premium freights in convergent supply chains. **Omega (United Kingdom)**, v. 80, p. 153–165, 2018.

AZADEH, A.; ASADZADEH, S. M.; TADAYOUN, S. Optimization of operator allocation in a large multi product assembly shop through unique integration of simulation and genetic algorithm. **International Journal of Advanced Manufacturing Technology**, v. 76, n. 1–4, p. 471–486, 2014.

AZADEH, A.; MOTEVALI HAGHIGHI, S.; ASADZADEH, S. M. A novel algorithm for layout optimization of injection process with random demands and sequence dependent setup times. **Journal of Manufacturing Systems**, v. 33, n. 2, p. 287–302, 2014.

AZIMI, P.; CHARMCHI, H. R. A new optimization via simulation approach for dynamic facility layout problem with budget constraints. **Modelling and Simulation in Engineering**, v. 2012, 2012.

BANDARU, S.; NG, A. H. C. **On the scalability of meta-models in simulation-based optimization of production systems**. 2015 Winter Simulation Conference (WSC). **Anais**...IEEE, dez. 2015Disponível em: <http://ieeexplore.ieee.org/document/7408523/>

BANKS, J. (ED.). **Handbook of simulation: Principles, methodology, advances, applications, and practice**. [s.l.] John Wiley & Sons, Inc., 1998.

BANKS, J. et al. **Discrete-Event System Simulation**. 5th. ed. [s.l.] Pearson, 2010.

BARLOW, E. et al. A mixed-method optimisation and simulation framework for supporting logistical decisions during offshore wind farm installations. **European Journal of Operational Research**, v. 264, n. 3, p. 894–906, 2018.

BARRA MONTEVECHI, J. A. et al. Conceptual modeling in simulation projects by mean adapted IDEF: An application in a Brazilian tech company. **Proceedings - Winter Simulation Conference**, n. 2008, p. 1624–1635, 2010.

BARTON, R. R. **Simulation optimization using metamodels**. Proceedings of the 2009 Winter Simulation Conference (WSC). **Anais**...IEEE, dez. 2009Disponível em: <http://ieeexplore.ieee.org/document/5429328/>

BERTRAND, J. W. M. .; FRANSOO, J. C. Operations management research methodologies using quantitative modeling. **International Journal of Operations & Production Management**, v. 22, n. 2, p. 241–264, fev. 2002.

BETTONVIL, B.; DEL CASTILLO, E.; KLEIJNEN, J. P. C. Statistical testing of optimality conditions in multiresponse simulation-based optimization. **European Journal of Operational Research**, v. 199, n. 2, p. 448–458, 2009.

BIANCHI, L. et al. A survey on metaheuristics for stochastic combinatorial optimization. **Natural Computing**, v. 8, n. 2, p. 239–287, 2009.

BIERLAIRE, M. Simulation and optimization: A short review. **Transportation Research Part C: Emerging Technologies**, v. 55, p. 4–13, 2015.

BOOTH, A.; PAPAIOANNOU, D.; SUTTON, A. (EDS.). **Systematic Approaches to a Successful Literature Review**. [s.l.] SAGE Publications, 2012.

BRYMAN, A. **Research Methods and Organization Studies**. London: Taylor & Francis, 2003.

BUITINCK, L. et al. API design for machine learning software: experiences from the scikit-learn project. **European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases**, p. 1–15, 2013.

BURKE, E. K. et al. Hyper-heuristics: A survey of the state of the art. **Journal of the Operational Research Society**, v. 64, n. 12, p. 1695–1724, 2013.

CALVET, L. et al. Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. **Open Mathematics**, v. 15, n. 1, p. 261–280, 2017.

CAN, B.; HEAVEY, C. Comparison of experimental designs for simulation-based symbolic regression of manufacturing systems. **Computers and Industrial Engineering**, v. 61, n. 3, p. 447–462, 2011.

CAN, B.; HEAVEY, C. A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models. **Computers and Operations Research**, v. 39, n. 2, p. 424–436, 2012.

CARSON, Y.; MARIA, A. Simulation Optimization: Methods and Applications. **Proceedings of the 29th conference on Winter simulation (1997)**, p. 118–126, 1997.

CENTRE FOR REVIEWS AND DISSEMINATION. **Systematic reviews: CRD's guidance for undertaking reviews in health care**. [s.l: s.n.].

CHEN, C.-H.; JIA, Q.-S.; LEE, L. H. (EDS.). **Stochastic Simulation Optimization for Discrete Event Systems**. 1st. ed. [s.l.] World Scientific, 2013.

CHICA, M. et al. Why Simheuristics? Benefits, Limitations, and Best Practices When Combining Metaheuristics with Simulation. **Ssrn**, 2017.

CHOI, C.; SEO, K.-M.; KIM, T. G. DEXSim: an experimental environment for distributed execution of replicated simulators using a concept of single simulation multiple scenarios. **Simulation**, v. 90, n. 4, p. 355–376, 2014.

CHWIF, L.; PAUL, R. J.; BARRETTO, M. R. P. Discrete event simulation model reduction: A causal approach. **Simulation Modelling Practice and Theory**, v. 14, n. 7, p. 930–944, 2006.

COLLIER, N.; NORTH, M. Parallel agent-based simulation with Repast for High Performance Computing. **Simulation**, v. 89, n. November, p. 1215–1235, 2012.

COSTA, A. et al. A parallel tabu search for solving the primal buffer allocation problem in serial production systems. **Computers and Operations Research**, v. 64, p. 97–112, 2015.

COSTA, E.; SOARES, A. L.; DE SOUSA, J. P. Information, knowledge and collaboration management in the internationalisation of SMEs: A systematic literature review. **International Journal of Information Management**, v. 36, n. 4, p. 557–569, 2016.

DAGKAKIS, G.; HEAVEY, C. A review of open source discrete event simulation software for operations research. **Journal of Simulation**, v. 10, n. 3, p. 1–14, 2016.

DELLINO, G.; MELONI, C. (EDS.). **Uncertainty Management in Simulation-Optimization of Complex Systems**. Boston, MA: Springer US, 2015. v. 59

DENYER, D.; TRANFIELD, D. **Producing a Systematic ReviewThe SAGE**

Handbook of Organizational Research Methods, 2009.

DENYER, D.; TRANFIELD, D.; VAN AKEN, J. E. Developing Design Propositions through Research Synthesis. **Organization Studies**, v. 29, n. 3, p. 393–413, 2008.

DORIGATTI, M. et al. A service-oriented framework for agent-based simulations of collaborative supply chains. **Computers in Industry**, v. 83, p. 92–107, 2016.

ERNST, A. T. et al. Staff scheduling and rostering: A review of applications, methods and models. **European Journal of Operational Research**, v. 153, n. 1, p. 3–27, 2004.

ESSAID, M. et al. GPU parallelization strategies for metaheuristics: a survey. **International Journal of Parallel, Emergent and Distributed Systems**, v. 5760, p. 1–26, 25 jan. 2018.

FRANCESCHINI, F.; MAISANO, D.; MASTROGIACOMO, L. Scientific journal publishers and omitted citations in bibliometric databases: Any relationship? **Journal of Informetrics**, v. 8, n. 3, p. 751–765, 2014.

FREITAG, M.; HILDEBRANDT, T. Automatic design of scheduling rules for complex manufacturing systems by multi-objective simulation-based optimization. **CIRP Annals - Manufacturing Technology**, v. 65, n. 1, p. 433–436, 2016.

FU, M. C. A tutorial review of techniques for simulation optimization. **Simulation Conference Proceedings, 1994. Winter**, p. 149–156, 1994a.

FU, M. C. Optimization via simulation: A review. **Annals of Operations Research**, v. 53, n. 1, p. 199–247, 1994b.

FU, M. C. Optimization for simulation: Theory vs. practice. **INFORMS Journal on Computing**, v. 14, n. 3, p. 192–215, 2002.

FU, M. C. (ED.). **Handbook of Simulation Optimization**. New York, NY: Springer New York, 2015. v. 216

FUJIMOTO, R. M. Research Challenges in Parallel and Distributed Simulation. **ACM Transactions on Modeling and Computer Simulation**, v. 26, n. 4, p. 1–29, 2016.

FUJIMOTO, R. M. et al. **Parallel discrete event simulation: The making of a field**. 2017 Winter Simulation Conference (WSC). **Anais**...IEEE, dez. 2017Disponível em: <http://ieeexplore.ieee.org/document/8247793/>

GAVRILAS, M. Heuristic and metaheuristic optimization techniques with application to power systems. **12th WSEAS International Conference on Mathemetical Methods and Computational Techniques in Electrical Engineering (MMACTEE'10), Selected Topics in Mathematical Methods and Computational Techniques in Electrical Engineering**, n. October 2010, p. 95–103, 2010.

GEYIK, F.; DOSDOĞRU, A. T. Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: an optimization via simulation approach. **Neural Computing and Applications**, v. 23, n. 6, p. 1631–1641, 2012.

GIGERENZER, G.; GAISSMAIER, W. Heuristic Decision Making. **Ssrn**, 2010.

GRULER, A. et al. A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. **International Transactions in Operational Research**, v. 00, p. 1–22, 2018.

GUTIN, G.; YEO, A.; ZVEROVICH, A. Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP. **Discrete Applied Mathematics**, v. 117, n. 1–3, p. 81–86, 2002.

HADJSAID, N. et al. Modeling cyber and physical interdependencies - Application in ICT and power grids. **2009 IEEE/PES Power Systems Conference and Exposition, PSCE 2009**, p. 1–6, 2009.

HAMMERSLEY, M. On ' Systematic ' Reviews of Research Literatures : A ' narrative ' response to Evans On ' Systematic ' Reviews of Research Literatures : a ' narrative ' response to Evans & Bene eld. **British Educational Research Journal**, v. 27, n. 5, p. 543–554, 2001.

HE, Y. et al. Improved exact and meta-heuristic methods for minimizing makespan of large-size SMSP. **Chemical Engineering Science**, v. 158, n. October 2016, p. 359–369, 2017.

HERRMANN, F. Simulation based priority rules for scheduling of a flow shop with simultaneously loaded stations. **Proceedings - 27th European Conference on Modelling and Simulation, ECMS 2013**, p. 775–781, 2013.

HILDEBRANDT, T.; GOSWAMI, D.; FREITAG, M. Large-scale simulation-based optimization of semiconductor dispatching rules. **Proceedings - Winter Simulation Conference**, v. 2015–Janua, p. 2580–2590, 2015.

HILLIER, F. S.; LIEBERMAN, G. J. **Introduction to Operational Research**. 10th. ed. [s.l.] McGraw-Hill, 2015.

ILES, M.; DEUGO, D.; CANADA, O. A Search for Routing Strategies in a Peer-to-Peer Network Using Genetic Programming. **Network**, p. 341–346, 2002.

JACKSON, I.; TOLUJEVS, J.; REGGELIN, T. The Combination of Discrete-Event Simulation and Genetic Algorithm for Solving the Stochastic Multi-Product Inventory Optimization Problem. **Transport and Telecommunication Journal**, v. 19, n. 3, p. 233–243, 2018.

JAFER, S.; LIU, Q.; WAINER, G. Synchronization methods in parallel and distributed discrete-event simulation. **Simulation Modelling Practice and Theory**, v. 30, p. 54–73, jan.

2013.

JAHANGIRIAN, M. et al. Simulation in manufacturing and business: A review. **European Journal of Operational Research**, v. 203, n. 1, p. 1–13, 2010.

JIA, S. et al. Improving performance of dispatch rules for daily scheduling of assembly and test operations. **Computers and Industrial Engineering**, v. 90, p. 86–106, 2015.

JUAN, A. A. et al. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. **Operations Research Perspectives**, v. 2, p. 62–72, 2015.

K.T., V.; PRABAGARAN, S.; JOSEPH, O. A. Dynamic due date assignment method: A simulation study in a job shop with sequence-dependent setups. **Journal of Manufacturing Technology Management**, 2017.

KELTON, W. D.; SADOWSKI, R. P.; SWETS, N. B. **Simulation with Arena**. 5th. ed. [s.l.] McGraw-Hill, 2010.

KILMER, R. A.; SMITH, A. E.; SHUMAN, L. J. Computing confidence intervals for stochastic simulation using neural network metamodels. **Computers & Industrial Engineering**, v. 36, n. 2, p. 391–407, 1999.

KIRK, D.; HWU, W.-M. W. **Programming Massively Parallel Processors: A Hands-on Approach**. Second ed. [s.l.] Elsevier, 2010.

KLEIJNEN, J. P. C. Kriging metamodeling in simulation: A review. **European Journal of Operational Research**, v. 192, n. 3, p. 707–716, 2009.

KLEIJNEN, J. P. C. Regression and Kriging metamodels with their experimental designs in simulation: A review. **European Journal of Operational Research**, v. 256, n. 1, p. 1–16, 2017.

KOVALCHUK, S. V. et al. Simulation of patient flow in multiple healthcare units using process and data mining techniques for model identification. **Journal of Biomedical Informatics**, v. 82, n. March 2017, p. 128–142, 2018.

KRAUSE, W.; SCHUTTE, C. DEVELOPING DESIGN PROPOSITIONS FOR AN OPEN INNOVATION APPROACH FOR SMEs. **South African Journal of Industrial Engineering**, v. 27, n. 3, p. 37–49, nov. 2016.

KUBAT, M. **An Introduction to Machine Learning**. Second ed. Cham: Springer International Publishing, 2017.

LAMIRI, M.; GRIMAUD, F.; XIE, X. Optimization methods for a stochastic surgery planning problem. **International Journal of Production Economics**, v. 120, n. 2, p. 400–410, ago. 2009.

LANDA, P. et al. Multiobjective bed management considering emergency and elective patient flows. **International Transactions in Operational Research**, v. 25, n. 1, p. 91–110, 2018.

LAROQUE, C. et al. **Fast converging, automated experiment runs for material flow simulations using distributed computing and combined metaheuristics**. Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC). **Anais**...IEEE, dez. 2012Disponível em: <http://ieeexplore.ieee.org/document/6465058/>

LAW, A. M.; KELTON, W. D. **Simulation Modeling and Analysis**. 2. ed. New York: McGraw-Hill, 1991. v. 2nd

LEARMONTH, M.; HARDING, N. EVIDENCE-BASED MANAGEMENT: THE VERY IDEA. **Public Administration**, v. 84, n. 2, p. 245–266, jun. 2006.

LEUNG, C. S. K.; LAU, H. Y. K. A hybrid multi-objective AIS-based algorithm applied to simulation-based optimization of material handling system. **Applied Soft Computing Journal**, v. 71, p. 553–567, 2018.

LI, S.; JIA, Y.; WANG, J. A discrete-event simulation approach with multiple-comparison procedure for stochastic resource-constrained project scheduling. **International Journal of Advanced Manufacturing Technology**, v. 63, n. 1–4, p. 65–76, 2012.

LI, W.; ÖZCAN, E.; JOHN, R. Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. **Renewable Energy**, v. 105, p. 473–482, maio 2017.

LONG-FEI, W.; LE-YUAN, S. Simulation Optimization: A Review on Theory and Applications. **Acta Automatica Sinica**, v. 39, n. 11, p. 1957–1968, 2013.

LOPES, H. DOS S. et al. Scenario analysis of Brazilian soybean exports via discrete event simulation applied to soybean transportation: The case of Mato Grosso State. **Research in Transportation Business and Management**, v. 25, n. October, p. 66–75, 2017.

LUCIDI, S. et al. A Simulation-Based Multiobjective Optimization Approach for Health Care Service Management. **IEEE Transactions on Automation Science and Engineering**, v. 13, n. 4, p. 1480–1491, 2016.

MAASHI, M.; KENDALL, G.; ÖZCAN, E. Choice function based hyper-heuristics for multi-objective optimization. **Applied Soft Computing Journal**, v. 28, p. 312–326, 2015.

MARSCHAN-PIEKKARI, R.; WELCH, C. **Handbook of Qualitative Research Methods for International Business**. [s.l.] Edward Elgar Publishing, Inc., 2004. v. 36

MAYNARD, H. B.; HODSON, W. K. (EDS.). **Maynard´s Industrial Engineering Handbook**. [s.l.] McGraw-Hill, 2004.

METHLEY, A. M. et al. PICO, PICOS and SPIDER: a comparison study of specificity and sensitivity in three search tools for qualitative systematic reviews. **BMC Health Services Research**, v. 14, n. 1, p. 579, 2014.

MIGUEL, P. A. C. et al. **Metodologia de pesquisa em engenharia de produção e gestão de operações**. segunda ed. Rio de Janeiro: Elsevier, 2012.

MIRANDA, R. DE C. **Redução do Espaço de Busca em Problemas de Otimização via Simulação Utilizando Análise Envoltória de Dados e Arranjos Ortogonais de Taguchi**. [s.l.] Unifervidade Federal de Itajubá, 2015.

MITROFF, I. I. et al. On Managing Science in the Systems Age: Two Schemas for the Study of Science as a Whole Systems Phenomenon. **Interfaces**, v. 4, n. 3, p. 46–58, 1974.

MOKHTARI, H.; SALMASNIA, A. A Monte Carlo simulation based chaotic differential evolution algorithm for scheduling a stochastic parallel processor system. **Expert Systems with Applications**, v. 42, n. 20, p. 7132–7147, 2015.

MONTAGNA, S.; VIROLI, M.; ROLI, A. A framework supporting multi-compartment stochastic simulation and parameter optimisation for investigating biological system development. **Simulation**, v. 91, n. 7, p. 666–685, 2015.

MORRELL, K. The narrative of "evidence based" management: A polemic. **Journal of Management Studies**, v. 45, n. 3, p. 613–635, 2008.

MUJICA MOTA, M.; FLORES DE LA MOTA, I. (EDS.). **Applied Simulation and Optimization 2**. Cham: Springer International Publishing, 2017.

MÜLLER, A. C.; GUIDO, S. **Introduction to Machine Learning with Python**. [s.l.] O'Reilly, 2017.

MUTA, H. et al. A multi-objective genetic algorithm using intermediate features of simulations. **Proceedings - Winter Simulation Conference**, v. 2015–Janua, p. 793–804, 2015.

NAGESHWARANIYER, S. S. A mine-to-mill economic analysis model and spectral imaging-based tracking system for a copper mine. **Journal of the Southern African Institute of Mining and Metallurgy**, v. 118, n. 1, p. 7–14, 2018.

NAGESHWARANIYER, S. S.; SON, Y.-J.; DESSUREAULT, S. Simulation-based optimal planning for material handling networks in mining. **SIMULATION**, v. 89, n. 3, p. 330–345, 9 mar. 2013a.

NAGESHWARANIYER, S. S.; SON, Y.-J.; DESSUREAULT, S. **Simulation-based robust optimization for complex truck-shovel systems in surface coal mines**. 2013 Winter Simulations Conference (WSC). **Anais**...IEEE, dez. 2013bDisponível em: <http://ieeexplore.ieee.org/document/6721714/>

NASIRI, M. M.; YAZDANPARAST, R.; JOLAI, F. A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule. **International Journal of Computer Integrated Manufacturing**, v. 30, n. 12, p. 1239–1252, 2017.

NAWARA, G.; HASSANEIN, E. Solving the Job-Shop Scheduling Problem by Arena Simulation Software. **International Journal of Engineering Innovation & Research**, v. 2, n. 2, p. 161–166, 2013.

NEGAHBAN, A.; SMITH, J. S. Simulation for manufacturing system design and operation: Literature review and analysis. **Journal of Manufacturing Systems**, v. 33, n. 2, p. 241–261, 2014.

NGUYEN, A.-T.; REITER, S.; RIGO, P. A review on simulation-based optimization methods applied to building performance analysis. **Applied Energy**, v. 113, p. 1043–1058, 2014.

NGUYEN, S.; ZHANG, M.; TAN, K. C. **Adaptive charting genetic programming for dynamic flexible job shop scheduling**. Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18. **Anais**...New York, New York, USA: ACM Press, 2018Disponível em: <http://dl.acm.org/citation.cfm?doid=3205455.3205531>

NOCEDAL, J.; WRIGHT, S. J. **Numerical Optimization**. [s.l: s.n.].

NOORDHOEK, M. et al. A simulation–optimization approach for a service-constrained multi-echelon distribution network. **Transportation Research Part E: Logistics and Transportation Review**, v. 114, n. February, p. 292–311, 2018.

OLIVEIRA, J. B.; LIMA, R. S.; MONTEVECHI, J. A. B. Perspectives and relationships in Supply Chain Simulation: A systematic literature review. **Simulation Modelling Practice and Theory**, v. 62, p. 166–191, 2016.

PARK, H.; FISHWICK, P. A. A GPU-Based Application Framework Supporting Fast Discrete-Event Simulation. **SIMULATION**, v. 86, n. 10, p. 613–628, 22 out. 2010.

PAWLEWSKI, P.; GREENWOOD, A. **Process Simulation and Optimization in Sustainable Logistics and Manufacturing**. Cham: Springer International Publishing, 2014.

PEDREGOSA, F. et al. Scikit-learn Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PHANDEN, R. K.; SAHARAN, L. K.; ERKOYUNCU, J. A. Simulation based cuckoo search optimization algorithm for flexible job shop scheduling problem. **Proceedings of the International Conference on Intelligent Science and Technology**, p. 50–55, 2018.

PILBEAM, C.; ALVAREZ, G.; WILSON, H. The governance of supply networks: a

systematic literature review. **Supply Chain Management**, v. 17, n. 4, p. 358–376, 2012.

PINHO, T. M. et al. Routing and schedule simulation of a biomass energy supply chain through SimPy simulation package. **Applied Computing and Informatics**, 2018.

POETING, M. et al. **A combined simulation optimization framework to improve operations in parcel logistics**. 2017 Winter Simulation Conference (WSC). **Anais**...IEEE, dez. 2017Disponível em: <http://ieeexplore.ieee.org/document/8248063/>

PONSIGNON, T.; MÖNCH, L. Simulation-based performance assessment of master planning approaches in semiconductor manufacturing. **Omega**, v. 46, p. 21–35, jul. 2014.

POSPIESZNY, P.; CZARNACKA-CHROBOT, B.; KOBYLINSKI, A. An effective approach for software project effort and duration estimation with machine learning algorithms. **Journal of Systems and Software**, v. 137, p. 184–196, 2018.

POWELL, W. B. A unified framework for stochastic optimization. **European Journal of Operational Research**, v. 0, p. 1–27, 2018.

RAJWANI, T.; LIEDONG, T. A. Political activity and firm performance within nonmarket research: A review and international comparative assessment. **Journal of World Business**, v. 50, n. 2, p. 273–283, 2015.

RAMASUBRAMANIAN, K.; SINGH, A. **Machine Learning Using R**. Berkeley, CA: Apress, 2019.

RAO, S. S. **Engineering Optimization Theory and Practice**. Fourth Edi ed. [s.l.] John Wiley & Sons, Inc., 2009.

RASCHKA, S.; JULIAN, D.; HEARTY, J. **Python: Deeper Insights into Machine Learning**. [s.l.] Packt Publishing Ltd., 2016.

RASKA, P.; ULRYCH, Z. Comparison of modified Downhill Simplex and Differential Evolution with other selected optimization methods used for discrete event simulation models. **Procedia Engineering**, v. 100, n. January, p. 807–815, 2015.

RESENDE, M. G. C.; RIBEIRO, C. C. **Optimization by GRASP**. New York, NY: Springer New York, 2016.

RIBEIRO, M. H.; PLASTINO, A.; MARTINS, S. L. Hybridization of GRASP metaheuristic with data mining techniques. **Journal of Mathematical Modelling and Algorithms**, v. 5, n. 1, p. 23–41, 2006.

RILEY, L. A. Discrete-event simulation optimization: A review of past approaches and propositions for future direction. **Simulation Series**, v. 45, n. 11, p. 386–393, 2013.

ROSSER, P. S.; SOMMERFELD, J. T.; TINCHER, W. C. Discrete-Event Simulation of Trouser Manufacturing. **International Journal of Clothing Science and Technology**, v. 3,

n. 2, p. 18–31, 1991.

RUIZ, M. et al. Using simulation-based optimization in the context of IT service management change process. **Decision Support Systems**, v. 112, n. June, p. 35–47, 2018.

RUPP, K. **42-years-processor-trend-625x396**. Disponível em: <https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/>.

SAILER, A. et al. Optimizing the Task Allocation Step for Multi-Core Processors within AUTOSAR. **International Conference on Applied Electronics**, 2013.

SALAM, M. A.; KHAN, S. A. Simulation based decision support system for optimization A case of Thai logistics service provider. **Industrial Management & Data Systems**, v. 116, n. 2, p. 236–254, 2016.

SALIMI, S.; MAWLANA, M.; HAMMAD, A. Performance analysis of simulation-based optimization of construction projects using High Performance Computing. **Automation in Construction**, v. 87, n. January, p. 158–172, 2018.

SALVENDY, G. **Handbook of Industrial Engineering**. 3th. ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2001.

SAREMI, A. et al. Appointment scheduling of outpatient surgical services in a multistage operating room department. **International Journal of Production Economics**, v. 141, n. 2, p. 646–658, 2013.

SAVINIEC, L.; SANTOS, M. O.; COSTA, A. M. Parallel local search algorithms for high school timetabling problems. **European Journal of Operational Research**, v. 265, n. 1, p. 81–98, 2018.

SCI-KIT LEARNING. **Sci-kit learning**. Disponível em: <http://scikit-learn.org/stable/index.html>.

SHAHI, M. R. M. et al. Optimization using simulation and response surface methodology with an application on subway train scheduling. **International Transactions in Operational Research**, v. 23, n. 4, p. 797–811, jul. 2016.

SHAHZAD, A.; MEBARKI, N. Learning Dispatching Rules for Scheduling: A Synergistic View Comprising Decision Trees, Tabu Search and Simulation. **Computers**, v. 5, n. 1, p. 3, 2016.

SIFALERAS, A.; KONSTANTARAS, I. Variable neighborhood descent heuristic for solving reverse logistics multi-item dynamic lot-sizing problems. **Computers and Operations Research**, v. 78, p. 385–392, 2017.

SONG, J.; QIU, Y.; LIU, Z. Integrating Optimal Simulation Budget Allocation and Genetic Algorithm to Find the Approximate Pareto Patient Flow Distribution. **IEEE**

Transactions on Automation Science and Engineering, v. 13, n. 1, p. 149–159, 2015.

SÖRENSEN, K. Metaheuristics-the metaphor exposed. **International Transactions in Operational Research**, v. 22, n. 1, p. 3–18, 2015.

SOUSA JUNIOR, W. T. DE et al. Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. **Computers and Industrial Engineering**, v. 128, n. January, p. 526–540, 2019.

SOYKAN, B.; RABADI, G. A Hybrid Metaheuristic Algorithm for Multi-Objective Runway Scheduling Problem in Simulation-based Optimization. **MODSIM World 2016At: Virginia Beach/VA**, n. April, p. 1–11, 2016.

TAHA, H. A. **Operations Research: An Introduction**. 8th. ed. [s.l.] Pearson, 2007.

TANSKANEN, K. et al. Towards evidence-based management of external resources: Developing design propositions and future research avenues through research synthesis. **Research Policy**, v. 46, n. 6, p. 1087–1105, 2017.

TEKIN, E.; SABUNCUOGLU, I. Simulation optimization: A comprehensive review on theory and applications. **IIE Transactions (Institute of Industrial Engineers)**, v. 36, n. 11, p. 1067–1081, 2004.

THOMAS, D. B.; HOWES, L.; LUK, W. A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation. **the ACM/SIGDA International Symposium on Field Programmable Gate Arrays**, p. 63–72, 2009.

TIACCI, L. Coupling a genetic algorithm approach and a discrete event simulator to design mixed-model un-paced assembly lines with parallel workstations and stochastic task times. **International Journal of Production Economics**, v. 159, p. 319–333, 2015.

TORRACO, R. J. Writing Integrative Literature Reviews: Guidelines and Examples. **Human Resource Development Review**, v. 4, n. 3, p. 356–367, 2005.

TURRIONI, J. B.; MELLO, C. H. P. **Metodologia de pesquisa em engenharia de produção**. Disponível em: <http://www.carlosmello.unifei.edu.br/Disciplinas/Mestrado/PCM-10/Apostila-Mestrado/Apostila_Metodologia_Completa_2012.pdf>. Acesso em: 29 out. 2018.

UHLIG, T.; ROSE, O. Simulation-based optimization for groups of cluster tools in semiconductor manufacturing using simulated annealing. **Proceedings of the Winter Simulation Conference**, p. 1857–1868, dez. 2011.

UPADHYAY, S. P.; ASKARI-NASAB, H. Simulation and optimization approach for uncertainty-based short-term planning in open pit mines. **International Journal of Mining Science and Technology**, v. 28, n. 2, p. 153–166, 2018.

VAN BREEDAM, A. Comparing descent heuristics and metaheuristics for the vehicle

routing problem. **Computers and Operations Research**, v. 28, n. 4, p. 289–315, 2001.

VAN VOLSEM, S.; DULLAERT, W.; VAN LANDEGHEM, H. An Evolutionary Algorithm and discrete event simulation for optimizing inspection strategies for multi-stage processes. **European Journal of Operational Research**, v. 179, n. 3, p. 621–633, 2007.

VÉJAR, A.; CHARPENTIER, P. Generation of an adaptive simulation driven by product trajectories. **Journal of Intelligent Manufacturing**, v. 23, n. 6, p. 2667–2679, 2012.

VOSNIAKOS, G.-C.; TSIFAKIS, A.; BENARDOS, P. Neural network simulation metamodels and genetic algorithms in analysis and design of manufacturing cells. **The International Journal of Advanced Manufacturing Technology**, v. 29, n. 5, p. 541–550, 2005.

WANG, S. et al. Towards smart factory for Industry 4.0: A self-organized multi-agent system with big data based feedback and coordination. **Computer Networks**, v. 101, p. 158–168, 2015.

XU, J. et al. Simulation Optimization: A Review and Exploration in the New Era of Cloud Computing and Big Data. **Asia-Pacific Journal of Operational Research**, v. 32, n. 03, p. 1550019, 2015.

XU, J. et al. Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. **Journal of Simulation**, v. 10, n. 4, p. 310–320, 2016.

YANG, T.; KUO, Y.; CHO, C. A genetic algorithms simulation approach for the multi-attribute combinatorial dispatching decision problem. **European Journal of Operational Research**, v. 176, n. 3, p. 1859–1873, 2007.

YOUSEFI, M. et al. Chaotic genetic algorithm and Adaboost ensemble metamodeling approach for optimum resource planning in emergency departments. **Artificial Intelligence in Medicine**, v. 84, p. 23–33, 2018.

ZSCHIESCHANG, E. et al. Resource efficiency-oriented optimization of material flow networks in chemical process engineering. **Procedia CIRP**, v. 15, p. 373–378, 2014.

ZÚÑIGA, E. R.; MORIS, M. U.; SYBERFELDT, A. **Integrating simulation-based optimization, lean, and the concepts of industry 4.0**. 2017 Winter Simulation Conference (WSC). **Anais**...IEEE, dez. 2017Disponível em: <http://ieeexplore.ieee.org/document/8248094/>

ZÚÑIGA, E. R.; SYBERFELDT, A.; MORIS, M. U. The Internet of Things, Factory of Things and Industry 4.0 in manufacturing: Current and future implementations. **Advances in Transdisciplinary Engineering**, v. 6, p. 221–226, 2017.

Apêndice I - Código para geração do espaço solução do OE1

```
"""
Código gerado em Python com as dependências definidas no Capítulo 4
"""
import random
import simpy
import time
from multiprocessing import Pool
import math

semente_randomica = 42 #para garantir a repetitibilidade entre vários cenários de otimização
media = 10.0          # Média do processamento em minutos
sigma = 2.0           # Sigma do processamento em minutos
parametro_quebra = 300.0            # Tempo médio para falha
media_quebra = 1 / parametro_quebra  # Variável de quebra
tempo_para_reparo = 30.0     # TEmpo const. para concertar uma maquina
tempo_outra_tarefa = 30.0    # Tempo para terminar outras tarefas

class maquina(object):
    def __init__(self, env, nome, pess_manuten):
        self.env = env
        self.nome = nome
        self.pecas_produzidas = 0
        self.quebra = False
        self.processo = env.process(self.reparar_maquina(pess_manuten))
        env.process(self.quebra_da_maquina())

    def reparar_maquina(self, pess_manuten):
        while True:
            tempo_produto = tempo_por_peca()
            while tempo_produto:
                try:
                    quando_comecou = self.env.now
                    yield self.env.timeout(tempo_produto)
                    tempo_produto = 0

                except simpy.Interrupt:
                    self.quebra = True
                    tempo_produto -= self.env.now - quando_comecou

                    with pess_manuten.request(priority=1) as req:
                        yield req
                        yield self.env.timeout(tempo_para_reparo)

                    self.quebra = False
            self.pecas_produzidas += 1

    def quebra_da_maquina(self):
```

```
        while True:
            yield self.env.timeout(tempo_para_falha())
            if not self.quebra:
                self.processo.interrupt()

def tempo_por_peca():
    return random.normalvariate(media, sigma)

def tempo_para_falha():
    return random.expovariate(media_quebra)

def outro_trabalho(env, pess_manuten):
    """The pess_manuten's other (unimportant) job."""
    while True:
        # quando_comecou a new job
        tempo_produto = tempo_outra_tarefa
        while tempo_produto:
            with pess_manuten.request(priority=2) as req:
                yield req
                try:
                    quando_comecou = env.now
                    yield env.timeout(tempo_produto)
                    tempo_produto = 0
                except simpy.Interrupt:
                    tempo_produto -= env.now - quando_comecou

def chama_o_programa (w):
    quantidade_de_manutencao = math.floor(w/1000000) #primeiro parâmetro
    NUM_maquinaS = (math.floor(w/1000)-(math.floor(w/1000000)*1000)) #segundo
parâmetro
    tempo_simulacao = ((w%1000)*10080) #terceiro parâmetro, computado em semanas e
trnformado em segundos
    random.seed(semente_randomica)  # Essa semente serve para repetir os resultados das
rodadas
    env = simpy.Environment()
    pess_manuten = simpy.PreemptiveResource(env, capcidade = quantidade_de_manutencao)
    maquinas = [maquina(env, 'maquina %d' % i, pess_manuten)
                for i in range(NUM_maquinaS)]
    env.process(outro_trabalho(env, pess_manuten))

    env.run(until=tempo_simulacao)

    total_de_pecas = 0
    for maquina in maquinas:
        total_de_pecas +=maquina.pecas_produzidas

    return ((total_de_pecas*100)-(quantidade_de_manutencao*20*tempo_simulacao)-
(tempo_simulacao*5))

def contrutor_de_expressao(a,b,c):
```

```
    return (a*1000000+b*1000+c)

if __nome__=="__main__":
    random.seed(semente_randomica) #para que todas os VETORES de simulação deêm o mesmo valor
    numero_de_nucleos = 24 #preferencialmente colocar valores multiplos de 4, pois em geral os núcleos são contados como esses múltiplos
    t1= time.time()
    posicao =0

    for re in range(1, 20):
        for ma in range(1, 40):
            for te in range(30, 150):
                posicao=re*1000000+ma*1000+te
                resultado=chama_o_programa(posicao)
                f=open("C:\\Users\\Lab-simul\\Documents\\saida.txt","a")
                f.write('%d ' '%d\n' %(posicao, resultado))
                f.close
            print ("\nValor atual de MAQUINA: ", ma, "!")
        print ("\nValor atual de REPARADOR: ", re, "!")
        print("tempo de processamento até o momento: ",math.floor( time.time()-t1), " segundos")
    print("tempo de processamento final: ",math.floor( time.time()-t1), " segundos")
```

Apêndice II - Código para geração do ótimo local do OE2

```
import time
from multiprocessing import Pool
import math
import itertools
import random
import simpy
import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt

def bubble_sort_inv(seq,res):
    changed = True
    while changed:
        changed = False
        for i in range(len(seq) - 1):
            if seq[i] < seq[i+1]:
                seq[i], seq[i+1] = seq[i+1], seq[i]
                res[i], res[i+1] = res[i+1], res[i]
                changed = True
    return None

def construtor_de_expressao(a,b,c):
    return (a*1000000+b*1000+c)

def divisor_de_expressao(d):
    return ((math.floor(d/1000000)),(math.floor(d/1000)-
(math.floor(d/1000000)*1000)),(d%1000))


def chama_o_programa (RANDOM_SEED, tamanho_estoque, nivel_seguranca,
Ver_est_min):
    SIM_TIME = 6912000#2592000  # Simulação em segundos => 8h p/ dia, 20 dias p/ mês e
12 meses

    def cliente(nome, env, estoque, nivel_estoque, espera_total):
        with estoque.request() as req:
            comeco = env.now
            yield req

            quantidade_estoque = 6+random.normalvariate (4,1)
            quantidade_requerida = int(quantidade_estoque)
            yield nivel_estoque.get(quantidade_requerida)
            tempo_ressuprimento = 3+random.normalvariate (2,1)
            yield env.timeout(abs(quantidade_requerida / (tempo_ressuprimento)))
            espera = env.now - comeco - (quantidade_requerida / tempo_ressuprimento)
```

```
            if espera > 0:
                yield espera_total.put(espera)

    def estoque_control(env, nivel_estoque):
        while True:
            if nivel_estoque.level / nivel_estoque.capacity * 100 < nivel_seguranca:
                yield env.process(transporte_ressuprimento(env, nivel_estoque))
            yield env.timeout(Ver_est_min)

    def transporte_ressuprimento(env, nivel_estoque):
        transporte_ressuprimento_tempo = (271+random.expovariate (10)-random.expovariate
(2))*tamanho_estoque
        yield env.timeout(int(transporte_ressuprimento_tempo))
        ammount = nivel_estoque.capacity - nivel_estoque.level
        yield nivel_estoque.put(ammount)

    def chegada_clientes(env, estoque, nivel_estoque, espera_total):
        for i in itertools.count():
            tempo_chegadas = random.normalvariate (36,10)-random.expovariate (2)+432
            yield env.timeout (int(tempo_chegadas))
            env.process(cliente('Cliente %d' % i, env, estoque, nivel_estoque, espera_total))

    env = simpy.Environment()
    estoque = simpy.Resource(env, 2)
    nivel_estoque = simpy.Container(env, tamanho_estoque, init=tamanho_estoque)
    espera_total = simpy.Container(env, capacity=99999999999, init=0)
    env.process(estoque_control(env, nivel_estoque))
    env.process(chegada_clientes(env, estoque, nivel_estoque, espera_total))
    env.run(until=SIM_TIME)
    b = espera_total.level
    return b

a=0
b=0
c=0
replicacao = 10

for loop_longo in range (9999999999999999):

    primeiro_parametro = random.randint(100,5000) #Tamnho do estoque de reposição
    segundo_parametro = random.randint(5,95) #Nível de segurança
    terceiro_parametro = 1800+(300*random.randint(0,89)) #Tempo para verificação do
estoque

    a=0
    for w in range (replicacao):

b=chama_o_programa(w+10,primeiro_parametro,segundo_parametro,terceiro_parametro)
#Semente rand, tamanho do lote, nível de segurança, periodo de revisão
        a+=b
```

```
    print ('Estamos no teste de numero: ', loop_longo)
    f=open("C:\\Users\\Lab-simul\\Documents\\saida_GRASP_aleatorio.txt","a")
    f.write('%d ' '%d ' '%d ' '%d\n' %(primeiro_parametro, segundo_parametro,
terceiro_parametro, a/replicacao))
    f.close
```

Apêndice III - Código para geração do ótimo local do OE3

```
import time
from multiprocessing import Pool
import math
import itertools
import random
import simpy

import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt
from numpy.random import RandomState

def chama_o_programa
(RNG_SEED,capacidade_M1,capacidade_M2,capacidade_M3,capacidade_M4,capacidade_
M5,capacidade_M6,capacidade_M7):
    ARR_RATE = 0.4
    taxa_M1 = 20.000
    taxa_M3 = 30.000
    taxa_M5 = 12.000
    taxa_M7 = 15.000

    def gerador_produto(env, arr_stream, arr_rate, atraso_inicial=0,
                        tempo_parada=simpy.core.Infinity, prng=RandomState(0)):

        produtos_criados = 0

        yield env.timeout(atraso_inicial)

        while env.now < tempo_parada:

            iat = prng.exponential(1.0 / arr_rate)

            tempo_M1 = prng.exponential(taxa_M1)
            tempo_M2 = 0.00001
            tempo_M3 = prng.exponential(taxa_M3)
            tempo_M4 = 0.00001
            tempo_M5 = prng.exponential(taxa_M5)
            tempo_M6 = 0.00001
            tempo_M7 = prng.exponential(taxa_M7)

            # Create new patient process instance
            produtos_criados += 1
            obp = produto_fluxo(env, 'Produto{}'.format(produtos_criados),
```

```
                tempo_M1=tempo_M1, tempo_M2=tempo_M2, tempo_M3=tempo_M3,
tempo_M4=tempo_M4, tempo_M5=tempo_M6, tempo_M6=tempo_M6,
tempo_M7=tempo_M7)

        env.process(obp)

        yield env.timeout(iat)

    def produto_fluxo(env, name, tempo_M1, tempo_M2, tempo_M3, tempo_M4, tempo_M5,
tempo_M6, tempo_M7):

        requerer_recurso = env.now
        requerer_recurso1 = M1_unidade.request()
        yield requerer_recurso1
        yield env.timeout(tempo_M1) # Stay in M1 bed

        requerer_recurso = env.now
        requerer_recurso2 = M2_unidade.request()
        yield requerer_recurso2

        M1_unidade.release(requerer_recurso1)

        yield env.timeout(tempo_M2) # Stay in M2 bed
        requerer_recurso = env.now
        requerer_recurso3 = M3_unidade.request()
        yield requerer_recurso3

        M2_unidade.release(requerer_recurso2)

        yield env.timeout(tempo_M3) # Stay in M2 bed
        requerer_recurso = env.now
        requerer_recurso4 = M4_unidade.request()
        yield requerer_recurso4

        M3_unidade.release(requerer_recurso3)

        yield env.timeout(tempo_M4) # Stay in M2 bed
        requerer_recurso = env.now
        requerer_recurso5 = M5_unidade.request()
        yield requerer_recurso5

        M4_unidade.release(requerer_recurso4)

        yield env.timeout(tempo_M5)
        requerer_recurso = env.now
        requerer_recurso6 = M6_unidade.request()
        yield requerer_recurso6

        M5_unidade.release(requerer_recurso5)
```

```
        yield env.timeout(tempo_M6)
        requerer_recurso = env.now
        requerer_recurso7 = M7_unidade.request()
        yield requerer_recurso7

        M6_unidade.release(requerer_recurso6)

        yield env.timeout(tempo_M7)
        M7_unidade.release(requerer_recurso7)
        if env.now>14400: # 10 dias de warmup
            total_que_saiu_do_sistemas.put(1)

    env = simpy.Environment()

    total_que_saiu_do_sistemas = simpy.Container(env, capacity=99999999999, init=0)

    prng = RandomState(RNG_SEED)

    # Declare Resources to model all units
    M1_unidade = simpy.Resource(env, capacidade_M1)
    M2_unidade = simpy.Resource(env, capacidade_M2)
    M3_unidade = simpy.Resource(env, capacidade_M3)
    M4_unidade = simpy.Resource(env, capacidade_M4)
    M5_unidade = simpy.Resource(env, capacidade_M5)
    M6_unidade = simpy.Resource(env, capacidade_M6)
    M7_unidade = simpy.Resource(env, capacidade_M7)

    runtime = 57600
    stop_arrivals = 57555
    env.process(gerador_produto(env, "Type1", ARR_RATE, 0, stop_arrivals, prng))
    env.run(until=runtime)
    return total_que_saiu_do_sistemas.level

replicacao = 10
loop_longo = 0
for primeiro_parametro in range (1,4):
    for segundo_parametro in range (1,11):
        for terceiro_parametro in range (1,4):
            for quarto_parametro in range (1,11):
                for quinto_parametro in range (1,4):
                    for sexto_parametro in range (1,11):
                        for setimo_parametro in range (1,4):
                            loop_longo+=1
                            a=0
                            b=0
                            for w in range (replicacao):

b=chama_o_programa(w+10,primeiro_parametro,segundo_parametro,terceiro_parametro,
quarto_parametro, quinto_parametro, sexto_parametro, setimo_parametro) #Semente rand,
tamanho do lote, nível de segurança, periodo de revisão
```

```
                    #print (b/10000)
                    a+=b
                    #print ('Estamos com D igual a: ', d)
                print ('Estamos no teste de numero: ', loop_longo)
                f=open("C:\\Users\\Wilson2018\\Desktop\\saida_EspSol_Law.txt","a")
                f.write('%d ' '%d ' '%d ' '%d ' '%d ' '%d ' '%d ' '%d\n' %(primeiro_parametro,
segundo_parametro, terceiro_parametro, quarto_parametro, quinto_parametro,
sexto_parametro, setimo_parametro, a/replicacao))
                f.close
```

APÊNDICE IV - Código de teste das AMs para os OE1 a 3

```
import pandas as pd
import math
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt
import time

df = pd.read_csv('df_2000_normalizado_cls_300_lvl.csv') #df vem da sigla dataframe

x_df=df[['reparador','maquinas','tempo']]
y_df=df['FO']

xDummies_df=pd.get_dummies(x_df)
yDummies_df=y_df

x = xDummies_df.values
y = yDummies_df.values

percentagem = 0.9 #parâmetro importante para se determinar a quantidade
#da massa de dados que será usada como treino e qual será usada como teste

tamanho_de_treino = math.floor(percentagem * len(y))
tamanho_de_teste = len(y) - tamanho_de_treino
#print ("Tamanho de treino: ",tamanho_de_treino)
#print ('Tamanho de teste: ',tamanho_de_teste)

treino_dados = x[:tamanho_de_treino]
treino_marcacoes = y [:tamanho_de_treino]

teste_dados = x[-tamanho_de_teste:]
teste_marcacoes = y[-tamanho_de_teste:]

def int_conf_para_a_media(data, confianca):
    a = 1.0*np.array(data)
    n = len(a)
    m, se = np.mean(a), scipy.stats.sem(a)
    h = se * sp.stats.t._ppf((1+confianca)/2., n-1)
    return m-h, m+h

def RESULTADOS(modelo,nome):

    print ("\nModelo: "+nome+" \n")

    t1= time.time()
    modelo.fit(treino_dados, treino_marcacoes)
    tempo_treino =(time.time()-t1)
```

```
    t1= time.time()
    resultado = modelo.predict(teste_dados)
    tempo_previsao =(time.time()-t1)

    diferencas = teste_marcacoes-resultado #vetor de diferença do real com o simulado
    IC95_i, IC95_f = int_conf_para_a_media(diferencas, 0.95)

    from sklearn.model_selection import cross_val_score
    k=3
    scores = cross_val_score(modelo, treino_dados, treino_marcacoes, cv=k)
    score3 = np.mean(scores)

    MAX = np.max(diferencas)
    MIN = np.min(diferencas)

    from sklearn import metrics
    MAE = metrics.mean_absolute_error(teste_marcacoes,resultado)
    MSE = metrics.mean_squared_error(teste_marcacoes,resultado)
    R2 = metrics.r2_score(teste_marcacoes,resultado)

    f=open("C:\\Users\\Lab-simula\\Desktop\\42 - i5\\saida.txt","a")
    f.write('%s, %.4f, %.4f, %.4f, %.4f, %.4f, %.4f, %.4f, %.4f \n' %(nome, score3, MIN, MAX,
MAE, MSE, R2, tempo_treino,tempo_previsao))
    f.close
    return resultado, diferencas

#1: MODELOS LINEARES GENERALIZADOS

#1.1) Ordinary Least Squares (OLS)
nome = "Ordinary Least Squares (OLS)"
from sklearn import linear_model
modelo = linear_model.LinearRegression()
OLS_res, OLS_dif = RESULTADOS(modelo,nome)

#1.2) Ridge Regression (RR)
nome = "Ridge Regression (RR)"
from sklearn import linear_model
modelo = linear_model.Ridge (alpha = .5)
RR_res, RR_dif = RESULTADOS(modelo,nome)

#1.3) Ridge Regression with Cross Validation (RRCV)
nome = "Ridge Regression with Cross Validation (RRCV)"
from sklearn import linear_model
modelo = linear_model.RidgeCV(alphas=[0.1, 1.0, 10.0])
RRCV_res, RRCV_dif = RESULTADOS(modelo,nome)

#1.4) Lasso (L)
nome = "Lasso (L)"
from sklearn import linear_model
```

```
modelo = linear_model.Lasso(alpha = 0.1)
L_res, L_dif = RESULTADOS(modelo,nome)

#1.5) Elastic Net (EN)
nome = "Elastic Net (EN)"
from sklearn.linear_model import ElasticNet
modelo = ElasticNet(alpha=0.1, l1_ratio=0.7)
EN_res, EN_dif = RESULTADOS(modelo,nome)

#1.7) Orthogonal Matching Pursuit (OMP)
nome = "Orthogonal Matching Pursuit(OMP)"
from sklearn.linear_model import OrthogonalMatchingPursuit
modelo = OrthogonalMatchingPursuit(n_nonzero_coefs=3)
OMP_res, OMP_dif = RESULTADOS(modelo,nome)

#1.8) Bayesian Ridge Regression (BRR)
nome = "Bayesian Ridge Regression (BRR)"
from sklearn import linear_model
modelo = linear_model.BayesianRidge()
BRR_res, BRR_dif = RESULTADOS(modelo,nome)

#1.9) Automatic Relevance Determination Regression (ARDR)
nome = "Automatic Relevance Determination Regression (ARDR)"
from sklearn.linear_model import ARDRegression
modelo = ARDRegression(compute_score=True)
ARDR_res, ARDR_dif = RESULTADOS(modelo,nome)

#1.10) Logistic Regression (LR)
nome = "Logistic Regression (LR)"
from sklearn.linear_model import LogisticRegression
modelo = LogisticRegression(C=100, penalty='l1', tol=0.01)
LR_res, LR_dif = RESULTADOS(modelo,nome)

#1.11) Stochastic Gradient Descent (SGD)
nome = "Stochastic Gradient Descent (SGD)"
from sklearn.linear_model import SGDClassifier
modelo = SGDClassifier(loss="hinge", penalty="l1",tol=0.01) #testar com squared_hinge
SGD_res, SGD_dif = RESULTADOS(modelo,nome)

#1.12) Polynomial Regression: extending linear models with basis functions (PR)
nome = "Polynomial Regression (PR)"
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

modelo = Pipeline([('poly', PolynomialFeatures(degree=3)),
                   ('linear', LinearRegression(fit_intercept=False))])
PR_res, PR_dif = RESULTADOS(modelo,nome)

#2. Análise Discriminante Linear e Quadratica
```

```
#2.1.) Linear Discriminant Analysis (LDA)
nome = "Linear Discriminant Analysis(LDA)"
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
modelo = LinearDiscriminantAnalysis(solver="svd", store_covariance=True)
#modelo = QuadraticDiscriminantAnalysis(store_covariances=True)
LDA_res, LDA_dif = RESULTADOS(modelo,nome)
#O modelo linear rodou, porém muito ruim, o quadrático não gerou nada.

#3. Kernel Ridge Regression (KRR)
nome = "Kernel Ridge Regression (KRR)"
from sklearn.kernel_ridge import KernelRidge
modelo = KernelRidge(alpha=1.0)
KRR_res, KRR_dif = RESULTADOS(modelo,nome)

#4. Support Vector Machines (SVM)
nome = "Support Vector Machines (SVM)"
from sklearn import svm
modelo = svm.SVC()
SVM_res, SVM_dif = RESULTADOS(modelo,nome)
#O SVM rodou muito ruim

#5. Stochastic Gradient Descent (SGD)
nome = "Stochastic Gradient Descent (SGD)"
from sklearn.linear_model import SGDClassifier
modelo = SGDClassifier(loss="hinge", penalty="l1")
SGD_res, SGD_dif = RESULTADOS(modelo,nome)
#O SGD foi muito ruim

#6. Nearest Neighbors Classification (NNC)
nome = "Nearest Neighbors Classification (NNC)"
from sklearn import neighbors
modelo = neighbors.KNeighborsClassifier(n_neighbors=15, weights='uniform')#em vez de
'uniform' testar 'distance'
NNC_res, NNC_dif = RESULTADOS(modelo,nome)

#7. Nearest Centroid Classifier (NCC)
nome = "Nearest Centroid Classifier (NCC)"
from sklearn.neighbors.nearest_centroid import NearestCentroid
modelo = NearestCentroid()
NCC_res, NCC_dif = RESULTADOS(modelo,nome)

#8. Gaussian Process Regressor (GPR)
nome = "Gaussian Process Regressor (GPR)"
#with noise-level estimation
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, WhiteKernel
kernel = 1.0 * RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e3)) \
        + WhiteKernel(noise_level=1e-5, noise_level_bounds=(1e-10, 1e+1))
```

```
#kernel = 1.0 * RBF(length_scale=100.0, length_scale_bounds=(1e-2, 1e3)) \
#         + WhiteKernel(noise_level=1, noise_level_bounds=(1e-10, 1e+1))
modelo = GaussianProcessRegressor(kernel=kernel,alpha=0.0)
GPR_res, GPR_dif = RESULTADOS(modelo,nome)

#10.1.  Gaussian Naive Bayes (GNB)
nome = "Gaussian Naive Bayes (GNB)"
from sklearn.naive_bayes import GaussianNB
modelo = GaussianNB()
GNB_res, GNB_dif = RESULTADOS(modelo,nome)

#10.2.  Multinomial Naive Bayes (MNB)
nome = "Multinomial Naive Bayes (MNB)"
from sklearn.naive_bayes import MultinomialNB
modelo = MultinomialNB()
MNB_res, MNB_dif = RESULTADOS(modelo,nome)

#10.3.  Bernoulli Naive Bayes (BNB)
nome = "Bernoulli Naive Bayes (BNB)"
from sklearn.naive_bayes import BernoulliNB
modelo = BernoulliNB()
BNB_res, BNB_dif = RESULTADOS(modelo,nome)

#11.1.  Decision Trees Classifier (DTC)
nome = "Decision Trees Classifier (DTC)"
from sklearn import tree
modelo = tree.DecisionTreeClassifier()
DTC_res, DTC_dif = RESULTADOS(modelo,nome)

#11.2.  Decision Trees Regressor (DTR)
nome = "Decision Trees Regressor (DTR)"
from sklearn import tree
modelo = tree.DecisionTreeRegressor()
DTR_res, DTR_dif = RESULTADOS(modelo,nome)

#12. Ensemble methods

#12.1. Bagging Classifier (BC)
nome = "Bagging Classifier (BC)"
from sklearn.ensemble import BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
modelo = BaggingClassifier(KNeighborsClassifier(),max_samples=0.5, max_features=0.5)
BC_res, BC_dif = RESULTADOS(modelo,nome)

#12.2. Random Forest Classifier (RFC)
nome = "Random Forest Classifier (RFC)"
from sklearn.ensemble import RandomForestClassifier
modelo = RandomForestClassifier(n_estimators=10)
RFC_res, RFC_dif = RESULTADOS(modelo,nome)
```

```
#12.3. Random Forest Regressor (RFR)
nome = "Random Forest Regressor (RFR)"
from sklearn.ensemble import RandomForestRegressor
modelo = RandomForestRegressor(max_depth=2, random_state=0)
RFR_res, RFR_dif = RESULTADOS(modelo,nome)

#12.4. Extra Trees Classifier (ETC)
nome = "Extra Trees Classifier (ETC)"
from sklearn.ensemble import ExtraTreesClassifier
modelo = ExtraTreesClassifier(n_estimators=10, max_depth=None, min_samples_split=2,
random_state=0)
ETC_res, ETC_dif = RESULTADOS(modelo,nome)

#12.5. Decision Tree Classifier (DTC)
nome = "Decision Tree Classifier (DTC)"
from sklearn.tree import DecisionTreeClassifier
modelo = DecisionTreeClassifier(max_depth=None, min_samples_split=2,random_state=0)
DTC_res, DTC_dif = RESULTADOS(modelo,nome)

#12.6. Ada Boost Classifier (ABC)
nome = "Ada Boost Classifier (ABC)"
from sklearn.ensemble import AdaBoostClassifier
modelo = AdaBoostClassifier(n_estimators=100)
ABC_res, ABC_dif = RESULTADOS(modelo,nome)

#12.7. Gradient Tree Boosting Regressor (GTBR)
nome = "Gradient Tree Boosting Regressor (GTBR)"
from sklearn.ensemble import GradientBoostingRegressor
modelo = GradientBoostingRegressor(n_estimators=100, learning_rate=1.0,
                                   max_depth=1, random_state=0)
GTBR_res, GTBR_dif = RESULTADOS(modelo,nome)

#13.Multiclass and multilabel algorithms

#13.1. One-Vs-Rest Classifier (OVRC)
nome = "One-Vs-Rest Classifier (OVRC)"
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
modelo = OneVsRestClassifier (LinearSVC(random_state=0))
GTBR_res, GTBR_dif = RESULTADOS(modelo,nome)

#13.2. One-Vs-One Classifier (OVOC)
nome = "One-Vs-One Classifier (OVOC)"
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import LinearSVC
modelo = OneVsOneClassifier (LinearSVC(random_state=0))
OVOC_res, OVOC_dif = RESULTADOS(modelo,nome)

#13.3. Error-Correcting Output-Codes (ECOC)
nome = "Error-Correcting Output-Codes (ECOC)"
```

```
from sklearn.multiclass import OutputCodeClassifier
from sklearn.svm import LinearSVC
modelo = OutputCodeClassifier(LinearSVC(random_state=0),code_size=2, random_state=0)
ECOC_res, ECOC_dif = RESULTADOS(modelo,nome)

#14. Neural network models

#14.1 Multi-layer Perceptron Classifier (MLPC)
nome = "Multi-layer Perceptron Classifier (MLPC)"
from sklearn.neural_network import MLPClassifier
modelo = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(15,),
                random_state=1)
MLPC_res, MLPC_dif = RESULTADOS(modelo,nome)
#muito ruim os resultados mais rodou

#14.1 Multi-layer Perceptron Regressor (MLPR)
nome = "Multi-layer Perceptron Regressor (MLPR)"
from sklearn.neural_network import MLPRegressor
modelo = MLPRegressor(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(15,),
                random_state=1)
MLPR_res, MLPR_dif = RESULTADOS(modelo,nome)

#Plota os Boxplot's dos resultados para a família de métodos MODELOS LINEARES
GENERALIZADOS
plt.figure()
data=(teste_marcacoes,OLS_res,RR_res,RRCV_res,L_res,EN_res,OMP_res,
    BRR_res,ARDR_res,LR_res,SGD_res,PR_res)
plt.ylabel('Range dos dados')
plt.xlabel('Métodos usados')
#plt.title('Boxplot real VS. previssões usadas')
plt.boxplot(data)
plt.show()

#Plota os Boxplot's da diferença entre real e simulado para a família de métodos MODELOS
LINEARES GENERALIZADOS
plt.ylabel('Range dos dados')
plt.xlabel('Métodos usados')
#plt.title('Boxplot das diferenças entre o real e o simulado')
data =(OLS_dif,RR_dif,RRCV_dif,L_dif,EN_dif,OMP_dif,
     BRR_dif,ARDR_dif,LR_dif,SGD_dif,PR_dif)
plt.boxplot(data)
plt.show()
#Plota os Boxplot's dos resultados PARA OS MÉTODOS RESTANTES
plt.figure()
data=(teste_marcacoes,LDA_res,KRR_res,SVM_res,SGD_res,NNC_res,NCC_res,GPR_res,
GNB_res,MNB_res,DTC_res,DTR_res,BC_res,RFC_res,RFR_res,ETC_res,DTC_res,
    ABC_res,GTBR_res,OVOC_res,ECOC_res,MLPC_res,MLPR_res)
plt.ylabel('Range dos dados')
plt.xlabel('Métodos usados')
#plt.title('Boxplot real VS. previssões usadas')
```

```
plt.boxplot(data)
plt.show()
#Plota os Boxplot's da diferença entre real e simulado PARA OS MÉTODOS RESTANTES
plt.ylabel('Range dos dados')
plt.xlabel('Métodos usados')
#plt.title('Boxplot das diferenças entre o real e o simulado')
data
=(LDA_dif,KRR_dif,SVM_dif,SGD_dif,NNC_dif,NCC_dif,GPR_dif,GNB_dif,MNB_dif,DT
C_dif,DTR_dif,BC_dif,RFC_dif,RFR_dif,ETC_dif,DTC_dif,
    ABC_dif,GTBR_dif,OVOC_dif,ECOC_dif,MLPC_dif,MLPR_dif)
plt.boxplot(data)
plt.show()
```

APÊNDICE V - Código integrando GA, AM e paralelismo ao OE1

```
import random
import simpy
import time
from multiprocessing import Pool
import math
import os
import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt

semente_randomica = 42 #para garantir a repetitibilidade entre vários cenários de otimização
media = 10.0          # Média do tempo de processamento
sigma = 2.0           # Sigma do tempo de processamento
parametro_quebra = 300.0            # Tempo médio para falha
media_quebra = 1 / parametro_quebra  # Variável de quebra
tempo_para_reparo = 30.0     # Tempo const. para concertar uma maquina
tempo_outra_tarefa = 30.0    # Tempo para terminar outras tarefas


#Ordena dois vetores de acordo com o primeiro vetor passado como parâmetro, o "seq"
#do maior para o menor valor, por isso é um método invertido, para garantir que o
#vetor da população terá em seus primeiros valores, os melhores indivíduos.
def bubble_sort_inv(seq,res):
    changed = True
    while changed:
        changed = False
        for i in range(len(seq) - 1):
            if seq[i] < seq[i+1]:
                seq[i], seq[i+1] = seq[i+1], seq[i]
                res[i], res[i+1] = res[i+1], res[i]
                changed = True
    return None

class maquina(object):
    def __init__(self, env, nome, pess_manuten):
        self.env = env
        self.nome = nome
        self.pecas_produzidas = 0
        self.quebra = False
        self.processo = env.process(self.reparar_maquina(pess_manuten))
        env.process(self.quebra_da_maquina())

    def reparar_maquina(self, pess_manuten):
```

```
        while True:
            tempo_produto = tempo_por_peca()
            while tempo_produto:
                try:
                    quando_comecou = self.env.now
                    yield self.env.timeout(tempo_produto)
                    tempo_produto = 0

                except simpy.Interrupt:
                    self.quebra = True
                    tempo_produto -= self.env.now - quando_comecou

                    with pess_manuten.request(priority=1) as req:
                        yield req
                        yield self.env.timeout(tempo_para_reparo)

                    self.quebra = False
            self.pecas_produzidas += 1

    def quebra_da_maquina(self):
        while True:
            yield self.env.timeout(tempo_para_falha())
            if not self.quebra:
                self.processo.interrupt()

def tempo_por_peca():
    return random.normalvariate(media, sigma)

def tempo_para_falha():
    return random.expovariate(media_quebra)

def outro_trabalho(env, pess_manuten):
    """The pess_manuten's other (unimportant) job."""
    while True:
        tempo_produto = tempo_outra_tarefa
        while tempo_produto:
            with pess_manuten.request(priority=2) as req:
                yield req
                try:
                    quando_comecou = env.now
                    yield env.timeout(tempo_produto)
                    tempo_produto = 0
                except simpy.Interrupt:
                    tempo_produto -= env.now - quando_comecou


def chama_o_programa (w):
    quantidade_de_manutencao = math.floor(w/1000000) #primeiro parâmetro
    NUM_maquinaS = (math.floor(w/1000)-(math.floor(w/1000000)*1000)) #segundo
parâmetro
```

```
   tempo_simulacao = ((w%1000)*10080) #terceiro parâmetro, computado em semanas e trnformado em segundos
   random.seed(semente_randomica)  # Essa semente serve para repetir os resultados das rodadas
   env = simpy.Environment()
   pess_manuten = simpy.PreemptiveResource(env, capcidade = quantidade_de_manutencao)
   maquinas = [maquina(env, 'maquina %d' % i, pess_manuten)
               for i in range(NUM_maquinaS)]
   env.process(outro_trabalho(env, pess_manuten))

   env.run(until=tempo_simulacao)

   total_de_pecas = 0
   for maquina in maquinas:
      total_de_pecas +=maquina.pecas_produzidas

   return ((total_de_pecas*100)-(quantidade_de_manutencao*20*tempo_simulacao)-(tempo_simulacao*5))

def contrutor_de_expressao(a,b,c):
   return (a*1000000+b*1000+c)

if __name__=="__main__":
   #Criação da solução inicial
   random.seed(semente_randomica) #para que todas os VETORES de simulação deêm o mesmo valor
   numero_de_nucleos = 120   #Tamanho máximo da população flutuante
   filhos_por_geracao = 1200  #Quantidade de filhos gerada por geração
   t1 = time.time()
   filhos = 40 #quantidades de filhos as quais serão calculados os valores reais para a geração futura
   vetor = [0]*numero_de_nucleos
   result = [0]*numero_de_nucleos
   result_y = [0]*numero_de_nucleos #para guardar so valores de "y" em função das CLASSES, para aprendizado dos métodos de ML
   temporario_vetor = [0]*filhos_por_geracao  #vetor para guardar os valores temporários das variaveis
   temporario_vetor_2 = [0]*filhos #guarda os valores em formato de cromossomo para passar como parâmetro para o POOL
   temporario_result = [0]*filhos_por_geracao #vetor para guardar os valores temporários das FOs
   db_variaveis = [0]*numero_de_nucleos # banco de dados para Machine Learn
   db_respostas = [0]*numero_de_nucleos # banco de respostas para Machine Learn

   #Cria a população inicial com uma variabilidade para representar o espaço solução, com 80 individuos
   #as outras 40 posições finais ficam em branco
   pop_constante=numero_de_nucleos-filhos_por_geracao
   for i in range(numero_de_nucleos):
      if i>=0 and i <=14: #Primeira metada = BAIXO, segunda metade = ALTO (AAA)
```

```
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=15 and i <=29: #(BAA)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=30 and i <=44: #(ABA)
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=45 and i <=59: #(AAB)
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=60 and i <=74: #(BBA)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=75 and i <=89: #(BAB)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=90 and i <=104: #(ABB)
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=105 and i <=119: #(BBB)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]

rotulos = ['manutenedor','maquinas','tempo'] #Rótulos para a lista dodb_variaveis
df = pd.DataFrame.from_records(db_variaveis, columns=rotulos)
```

```
x_df=df[['manutenedor','maquinas','tempo']]
xDummies_df=pd.get_dummies(x_df)
x = xDummies_df.values

p=Pool()
result = p.map(chama_o_programa, vetor) #roda com os números do vetor
p.close()
p.join()

for i in range (len(result)):
   result_y[i]=math.floor(result[i]/300000)


y=np.array(result_y) #Transforma o vetor de tipo normal em do tipo NumPy, com os resultados em CLASSES

cont=0 #Contador de interações sem melhora
while (cont<=11): #loop para verificar as interações sem melhora
   w=0
   for i in range(int(filhos_por_geracao/2)):     #cada vez que o loop roda, dois filhos são gerados
      a=random.randint(0,79)   #o primeiro pai, dos 80 possíveis
      b=random.randint(0,79)   #o segundo pai, dos 80 possíveis
      if a==b:
         while (a==b):
            b=random.randint(0,79)
      x1, x2, x3 = divisor_de_expressao(vetor[a])
      y1, y2, y3 = divisor_de_expressao(vetor[b])
      #print (i)
      #mode=int(raw_input('Input:'))

      #Operação de MUTAÇÃO
      mut_prob=8 #probabilidade de mutação tanto para o primeiro quanto para o segundo filho
      mutation = random.randint(1,100) #probabilidade para o PRIMEIRO filho sofrer mutação
      if mutation<=mut_prob:    #existe 8% de chance de ocorrer mutação
         c=random.randint(0,2) # qual dos 3 cromossomos será trocado na mutação
         if c==0:
            x1=random.randint(2,20) #troca do valor para o REPARADOR
         if c==1:
            x2=random.randint(2,40) #troca do valor para a MAQUINA
         if c==3:
            x3=random.randint(30,150) #troca do valor para o TEMPO

      mutation = random.randint(1,100) #probabilidade para o SEGUNDO filho sofrer mutação
      if mutation<=mut_prob:    #existe 8% de chance de ocorrer mutação
         c=random.randint(0,2) # qual dos 3 cromossomos será trocado na mutação
```

```
            if c==0:
               y1=random.randint(2,20) #troca do valor para o REPARADOR
            if c==1:
               y2=random.randint(2,40) #troca do valor para a MAQUINA
            if c==3:
               y3=random.randint(30,150) #troca do valor para o TEMPO

         # Operação de CROSSOVER
         c=random.randint(0,2) # qual dos 3 cromossomos será trocado no crossover

         if c==0:
            temporario_vetor[w]= [y1,x2,x3]
            temporario_vetor[w+1]= [x1,y2,y3]

         if c==1:
            temporario_vetor[w]= [x1,y2,x3]
            temporario_vetor[w+1]= [y1,x2,y3]

         if c==2:
            temporario_vetor[w]= [x1,x2,y3]
            temporario_vetor[w+1]= [y1,y2,x3]

         w+=2 #Implementa para os próximos 2 filhos

      #8. Gaussian Process Regressor (GPR)
      nome = "Gaussian Process Regressor (GPR)"
      #with noise-level estimation
      from sklearn.gaussian_process import GaussianProcessRegressor
      from sklearn.gaussian_process.kernels import RBF, WhiteKernel
      kernel = 1.0 * RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e3)) \
             + WhiteKernel(noise_level=1e-5, noise_level_bounds=(1e-10, 1e+1))
      #kernel = 1.0 * RBF(length_scale=100.0, length_scale_bounds=(1e-2, 1e3)) \
      #        + WhiteKernel(noise_level=1, noise_level_bounds=(1e-10, 1e+1))
      modelo = GaussianProcessRegressor(kernel=kernel,alpha=0.0)
      modelo.fit(x, y)
      temporario_result = modelo.predict(temporario_vetor)

      bubble_sort_inv(temporario_result,temporario_vetor)

      x=np.concatenate([x,temporario_vetor[:filhos]], axis=0)

      for i in range (len(temporario_vetor_2)): #Antes de chamar o POOL, juntar as variáveis
x1 x2 e x3
         temporario_vetor_2[i] =
contrutor_de_expressao(temporario_vetor[i][0],temporario_vetor[i][1],temporario_vetor[i][2]
)

      p=Pool()
      temporario_result[:filhos] = p.map(chama_o_programa, temporario_vetor_2[:filhos])
#roda com os números do vetor
```

```
    p.close()
    p.join()

    for i in range (len(temporario_result[:filhos])):  #Depois do POOL dar o APPEND no vetor Y
        y=np.append(y,math.floor(temporario_result[i]/300000)) #só numeros inteiros são passados

    for i in range(filhos):
        w=random.randint(5,119)
        vetor[w]=temporario_vetor_2[i]
        result[w]=temporario_result[i]

    antes=result[0]
    #Ordena do maior para o menor os vetores de Respontas e de Parâmetros,
    #em função das Resposta (que são as Fo's)
    bubble_sort_inv(result,vetor)

    depois=result[0]

    if (antes-depois)==0:
        cont+=1
    else:
        cont=0

    f=open("C:\\Users\\Lab-Simul\\Desktop\\saida.txt","a")
    f.write('%d ' '%d ' '%d ' '%d\n' %(result[0], vetor[0], cont, antes-depois, math.floor(time.time()-t1)))
    f.close
```

APÊNDICE VI - Código integrando GRASP, AM e paralelismo ao OE1

```
import random
import simpy
import time
from multiprocessing import Pool
import math
import os
import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt

semente_randomica = 42 #para garantir a repetitibilidade entre vários cenários de otimização
media = 10.0          # Média do tempo de processamento
sigma = 2.0           # Sigma do tempo de processamento
parametro_quebra = 300.0           # Tempo médio para falha
media_quebra = 1 / parametro_quebra  # Variável de quebra
tempo_para_reparo = 30.0     # Tempo const. para concertar uma maquina
tempo_outra_tarefa = 30.0    # Tempo para terminar outras tarefas

#Ordena dois vetores de acordo com o primeiro vetor passado como parâmetro, o "seq"
#do maior para o menor valor, por isso é um método invertido, para garantir que o
#vetor da população terá em seus primeiros valores, os melhores indivíduos.
def bubble_sort_inv(seq,res):
    changed = True
    while changed:
        changed = False
        for i in range(len(seq) - 1):
            if seq[i] < seq[i+1]:
                seq[i], seq[i+1] = seq[i+1], seq[i]
                res[i], res[i+1] = res[i+1], res[i]
                changed = True
    return None

class maquina(object):
    def __init__(self, env, nome, pess_manuten):
        self.env = env
        self.nome = nome
        self.pecas_produzidas = 0
        self.quebra = False
        self.processo = env.process(self.reparar_maquina(pess_manuten))
        env.process(self.quebra_da_maquina())

    def reparar_maquina(self, pess_manuten):
        while True:
```

```
            tempo_produto = tempo_por_peca()
            while tempo_produto:
                try:
                    quando_comecou = self.env.now
                    yield self.env.timeout(tempo_produto)
                    tempo_produto = 0

                except simpy.Interrupt:
                    self.quebra = True
                    tempo_produto -= self.env.now - quando_comecou

                    with pess_manuten.request(priority=1) as req:
                        yield req
                        yield self.env.timeout(tempo_para_reparo)

                    self.quebra = False
            self.pecas_produzidas += 1

    def quebra_da_maquina(self):
        while True:
            yield self.env.timeout(tempo_para_falha())
            if not self.quebra:
                self.processo.interrupt()

def tempo_por_peca():
    return random.normalvariate(media, sigma)

def tempo_para_falha():
    return random.expovariate(media_quebra)

def outro_trabalho(env, pess_manuten):
    """The pess_manuten's other (unimportant) job."""
    while True:
        tempo_produto = tempo_outra_tarefa
        while tempo_produto:
            with pess_manuten.request(priority=2) as req:
                yield req
                try:
                    quando_comecou = env.now
                    yield env.timeout(tempo_produto)
                    tempo_produto = 0
                except simpy.Interrupt:
                    tempo_produto -= env.now - quando_comecou

def chama_o_programa (w):
    quantidade_de_manutencao = math.floor(w/1000000) #primeiro parâmetro
    NUM_maquinaS = (math.floor(w/1000)-(math.floor(w/1000000)*1000)) #segundo
parâmetro
    tempo_simulacao = ((w%1000)*10080) #terceiro parâmetro, computado em semanas e
trnformado em segundos
```

```
   random.seed(semente_randomica)  # Essa semente serve para repetir os resultados das rodadas
   env = simpy.Environment()
   pess_manuten = simpy.PreemptiveResource(env, capcidade = quantidade_de_manutencao)
   maquinas = [maquina(env, 'maquina %d' % i, pess_manuten)
               for i in range(NUM_maquinaS)]
   env.process(outro_trabalho(env, pess_manuten))

   env.run(until=tempo_simulacao)

   total_de_pecas = 0
   for maquina in maquinas:
      total_de_pecas +=maquina.pecas_produzidas

   return ((total_de_pecas*100)-(quantidade_de_manutencao*20*tempo_simulacao)-(tempo_simulacao*5))

def contrutor_de_expressao(a,b,c):
   return (a*1000000+b*1000+c)

if __name__=="__main__":
   #Criação da solução inicial
   random.seed(semente_randomica) #para que todas os VETORES de simulação deêm o mesmo valor
   numero_de_nucleos = 120   #Tamanho máximo da população flutuante
   filhos_por_geracao = 40  #Quantidade de filhos gerada por geração
   t1= time.time()
   vetor = [0]*numero_de_nucleos
   result = [0]*numero_de_nucleos
   result_y = [0]*numero_de_nucleos #para guardar so valores de "y" em função das CLASSES, para aprendizado dos métodos de ML
   temporario_vetor = [0]*126  #vetor para guardar os valores temporários das variaveis
   temporario_vetor_2 = [0]*126 #guarda os valores para depois inserir na data-base
   temporario_result = [0]*126 #vetor para guardar os valores temporários das FOs
   temporario_result_2 = [0]*126
   temporario_result_total = []
   temporario_vetor_total = []
   lista_melhores_vetor = [-9999999999]*60 # Vetor para guardar as melhores combinações de cada iteração
   lista_melhores_result = [-9999999999]*60#guarda os resultados das melhores iterações
   db_variaveis = [0]*numero_de_nucleos # banco de dados para Machine Learn
   db_respostas = [0]*numero_de_nucleos # banco de respostas para Machine Learn

   #Cria a população inicial com uma variabilidade para representar o espaço solução, com 80 individuos
   #as outras 40 posições finais ficam em branco
   pop_constante=numero_de_nucleos-filhos_por_geracao
   for i in range(numero_de_nucleos):
      if i>=0 and i <=14: #Primeira metada = BAIXO, segunda metade = ALTO (AAA)
         a=random.randint(11,20)  #o range para o REPARADOR
```

```
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=15 and i <=29: #(BAA)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=30 and i <=44: #(ABA)
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=45 and i <=59: #(AAB)
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=60 and i <=74: #(BBA)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(90,150) #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=75 and i <=89: #(BAB)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(18,40)  #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=90 and i <=104: #(ABB)
        a=random.randint(11,20)  #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]
    if i>=105 and i <=119: #(BBB)
        a=random.randint(2,10)   #o range para o REPARADOR
        b=random.randint(2,17)   #o range para o número de MÁQUINAS
        c=random.randint(30,89)  #o range para a quantidade de TEMPO
        vetor[i]=contrutor_de_expressao(a,b,c)
        db_variaveis[i]=[a,b,c]

print ("\n Vetor do espaço solução inicial está abaixo com ",len(vetor)," elementos: \n")
print (vetor[:])
```

```
   rotulos = ['manutenedor','maquinas','tempo'] #Rótulos para a lista dodb_variaveis
   df = pd.DataFrame.from_records(db_variaveis, columns=rotulos)
   print (df)

   x_df=df[['manutenedor','maquinas','tempo']]
   xDummies_df=pd.get_dummies(x_df)
   x = xDummies_df.values

   p=Pool()
   result = p.map(chama_o_programa, vetor) #roda com os números do vetor
   p.close()
   p.join()

   for i in range (len(result)):
      result_y[i]=math.floor(result[i]/300000)

   y=np.array(result_y) #Transforma o vetor de tipo normal em do tipo NumPy, com os
resultados em CLASSES

   bubble_sort_inv(result,vetor)#Essa ordenação garante que os primeiros números
   cont=0 #Conta quantas iterações já foram feitas
   cont_iter=0 #Conta quantas interações foram feitas sem melhora
   #8. Gaussian Process Regressor (GPR)
   nome = "Gaussian Process Regressor (GPR)"
   from sklearn.gaussian_process import GaussianProcessRegressor
   from sklearn.gaussian_process.kernels import RBF, WhiteKernel
   kernel = 1.0 * RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e3)) \
          + WhiteKernel(noise_level=1e-5, noise_level_bounds=(1e-10, 1e+1))
   modelo = GaussianProcessRegressor(kernel=kernel,alpha=0.0)
   modelo.fit(x, y)

   while (cont_iter<=13): #Garante que no mínimo 10 iterações sem melhora foram geradas
         for hh in range (10):
            a=random.randint(0,95) # De 0 a 95 representam os 96 elementos que representam
80%
            b=random.randint(0,95) # do espaço solução original, um ALFA para a Lista Restrita
de
            c=random.randint(0,95) # Candidatos igual a 80%

            if a==b:                    # Só para garantir que os números serão diferentes
               while (a==b):            # assim os números não se repetirão
                  b=random.randint(0,95)
            if a==c:
               while (a==c):
                  c=random.randint(0,95)
            if b==c:
               while (b==c):
                  c=random.randint(0,95)

            sp1, x2, x3 = divisor_de_expressao(vetor[a]) # A Solução Parcial é criada aqui,
```

```
y1, sp2, y3 = divisor_de_expressao(vetor[b]) # formada por sp1, sp2 e sp3
z1, z2, sp3 = divisor_de_expressao(vetor[c])

# Início da fase de BUSCA do GRASP
#Definição da magnitude dos vizinhos em torno da PRIMEIRA variável
# de maneira tal que seja como: q1|q2|sp1|q4|q5
if (sp1-2)==0: # Define os valores de vizinhos para a variável Manutenedor
   q1=sp1
   q2=sp1
if (sp1-2)==1:
   q1=sp1-1
   q2=sp1-1
if (sp1-2)>=2:
   q1=sp1-2
   q2=sp1-1

if (sp1-20)==0:
   q4=sp1
   q5=sp1
if (sp1-20)==-1:
   q4=sp1+1
   q5=sp1+1
if (sp1-20)<=2:
   q4=sp1+1
   q5=sp1+2

#Definição da magnitude dos vizinhos em torno da SEGUNDA variável
# de maneira tal que seja como: w1|w2|sp2|w4|w5
if (sp2-2)==0: # Define os valores de vizinhos para a variável Maquinas
   w1=sp2
   w2=sp2
if (sp2-2)==1:
   w1=sp2-1
   w2=sp2-1
if (sp2-2)>=2:
   w1=sp2-2
   w2=sp2-1

if (sp2-40)==0:
   w4=sp2
   w5=sp2
if (sp2-40)==-1:
   w4=sp2+1
   w5=sp2+1
if (sp2-40)<=2:
   w4=sp2+1
   w5=sp2+2

#Definição da magnitude dos vizinhos em torno da TERCEIRA variável
# de maneira tal que seja como: e1|e2|sp3|e4|e5
```

```
            if (sp3-30)==0: # Define os valores de vizinhos para a variável Tempo
               e1=sp3
               e2=sp3
            if (sp3-30)==1:
               e1=sp3-1
               e2=sp3-1
            if (sp3-30)==2:
               e1=sp3-2
               e2=sp3-1
            if (sp3-30)==3:
               e1=sp3-3
               e2=sp3-2
            if (sp3-30)>=4:
               e1=sp3-4
               e2=sp3-2

            if (sp3-150)==0:
               e4=sp3
               e5=sp3
            if (sp3-150)==-1:
               e4=sp3+1
               e5=sp3+1
            if (sp3-150)==-2:
               e4=sp3+1
               e5=sp3+2
            if (sp3-150)==-3:
               e4=sp3+2
               e5=sp3+3
            if (sp3-150)<=4:
               e4=sp3+2
               e5=sp3+4

            Man = [q1,q2,sp1,q4,q5]
            Maq = [w1,w2,sp2,w4,w5]
            Tem = [e1,e2,sp3,e4,e5]

            marcador=0

            for i in range (len(Man)):
               for j in range (len(Maq)):
                  for k in range (len(Tem)):

temporario_vetor[marcador]=contrutor_de_expressao(Man[i],Maq[j],Tem[k])
                        temporario_vetor_2[marcador]=[Man[i],Maq[j],Tem[k]]

                        marcador+=1

            temporario_result_2 = modelo.predict(temporario_vetor_2[:125])

            bubble_sort_inv(temporario_result_2,temporario_vetor)
```

```
            for qq in range (40):
                temporario_result_total.append(temporario_result_2[qq])
                temporario_vetor_total.append(temporario_vetor[qq])

        ########################
        bubble_sort_inv(temporario_result_total,temporario_vetor_total)

        p=Pool()
        temporario_result[:40] = p.map(chama_o_programa, temporario_vetor_total[:40])
#roda com os números do vetor
        p.close()
        p.join()

        bubble_sort_inv(temporario_result[:40],temporario_vetor_total[:40])

        for i in range (len(temporario_result[:40])):
            temporario_vetor_2[i]=[divisor_de_expressao(temporario_vetor_total[i])]
            x=np.concatenate([x,temporario_vetor_2[i]], axis=0)

        for i in range (len(temporario_result[:40])):   #Depois do POOL dar o APPEND no vetor
Y
            y=np.append(y,math.floor(temporario_result[i]/300000)) #só numeros inteiros são
passados

        modelo.fit(x, y)
        lista_melhores_vetor[cont]=temporario_vetor_total[0]
        lista_melhores_result[cont]=temporario_result[0]

        antes=lista_melhores_result[0]
        bubble_sort_inv(lista_melhores_result,lista_melhores_vetor)
        depois = lista_melhores_result[0]

        if (antes-depois)==0:
            cont_iter+=1
        else:
            cont_iter=0
        cont+=1
        print ("Contador de iterações igual a: ", cont-1)
        print("Tempo de processamento SERIAL igual a: ",math.floor( time.time()-t1), "
segundos, até o momento")
        print ("\n Estamos à ",cont_iter," populações sem melhora na melhor solução \n")
```

APÊNDICE VII - Código integrando GRASP, AM e paralelismo ao OE2

```
import time
from multiprocessing import Pool
import math
import itertools
import random
import simpy
import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt


def bubble_sort_inv(seq,res):
    changed = True
    while changed:
        changed = False
        for i in range(len(seq) - 1):
            if seq[i] < seq[i+1]:
                seq[i], seq[i+1] = seq[i+1], seq[i]
                res[i], res[i+1] = res[i+1], res[i]
                changed = True
    return None

def construtor_de_expressao(a,b,c,d):
    return (a*100000000000+b*10000000+c*100000+d)

def divisor_de_expressao(e):
    return ((math.floor(e/100000000000)),(math.floor(e/10000000)-
(math.floor(e/100000000000)*10000)),(math.floor(e/100000)-
(math.floor(e/10000000)*100)),(e%100000))

def divisor_de_expressao_2(e):
    return ((math.floor(e/10000000)),(math.floor(e/100000)-
(math.floor(e/10000000)*100)),(e%100000))

def divisor_de_expressao_3(e):
    return ((math.floor(e/10000000)-
(math.floor(e/100000000000)*10000)),(math.floor(e/100000)-
(math.floor(e/10000000)*100)),(e%100000))

def chama_o_programa (RANDOM_SEED, tamanho_estoque, nivel_seguranca,
Ver_est_min):
    SIM_TIME = 6912000#2592000  # Simulação em segundos => 8h p/ dia, 20 dias p/ mês e
12 meses
```

```
    def cliente(nome, env, estoque, nivel_estoque, espera_total):
        with estoque.request() as req:
            comeco = env.now
            yield req

            quantidade_estoque = 6+random.normalvariate (4,1)
            quantidade_requerida = int(quantidade_estoque)
            yield nivel_estoque.get(quantidade_requerida)
            tempo_ressuprimento = 3+random.normalvariate (2,1)
            yield env.timeout(abs(quantidade_requerida / (tempo_ressuprimento)))
            espera = env.now - comeco - (quantidade_requerida / tempo_ressuprimento)
            if espera > 0:
                yield espera_total.put(espera)

    def estoque_control(env, nivel_estoque):
        while True:
            if nivel_estoque.level / nivel_estoque.capacity * 100 < nivel_seguranca:
                yield env.process(transporte_ressuprimento(env, nivel_estoque))
            yield env.timeout(Ver_est_min)

    def transporte_ressuprimento(env, nivel_estoque):
        transporte_ressuprimento_tempo = (271+random.expovariate (10)-random.expovariate
(2))*tamanho_estoque
        yield env.timeout(int(transporte_ressuprimento_tempo))
        ammount = nivel_estoque.capacity - nivel_estoque.level
        yield nivel_estoque.put(ammount)

    def chegada_clientes(env, estoque, nivel_estoque, espera_total):
        for i in itertools.count():
            tempo_chegadas = random.normalvariate (36,10)-random.expovariate (2)+432
            yield env.timeout (int(tempo_chegadas))
            env.process(cliente('Cliente %d' % i, env, estoque, nivel_estoque, espera_total))

    env = simpy.Environment()
    estoque = simpy.Resource(env, 2)
    nivel_estoque = simpy.Container(env, tamanho_estoque, init=tamanho_estoque)
    espera_total = simpy.Container(env, capacity=99999999999, init=0)
    env.process(estoque_control(env, nivel_estoque))
    env.process(chegada_clientes(env, estoque, nivel_estoque, espera_total))
    env.run(until=SIM_TIME)
    b = espera_total.level
    T_Esp = espera_total.level

    t1 = 1621*tamanho_estoque
    t2 = 154*T_Esp+1.8*nivel_seguranca-3.9*(Ver_est_min*Ver_est_min)
    t3 = 221000000*(tamanho_estoque**2)
    t4 = 83.4*tamanho_estoque*T_Esp
    t51 = round (T_Esp, 4)+1
    t5 = (0.22*(tamanho_estoque**(3/t51)))
```

```
    teste = round(0.011*(nivel_seguranca**2),8)

    FO = (-1149557-t1-t2+t4-t5-teste-t3)/45683000
    return round (FO,4)

if __name__=="__main__":
    # População inicial

    pop = 600

    vetor_populacao_identidade = [0]*pop
    vetor_populacao_resultado = [0]*pop
    db_variaveis = [0]*pop
    result = [0]*pop
    result_y = [0]*pop
    SEMENTE = 42
    x1 = SEMENTE #semente randômica
    t1= time.time()

    temporario_vetor = [0]*126  #vetor para guardar os valores temporários das variaveis
    temporario_vetor_2 = [0]*126 #guarda os valores para depois inserir na data-base
    temporario_result = [0]*126 #vetor para guardar os valores temporários das FOs
    temporario_result_2 = [0]*126
    temporario_result_total = []
    temporario_vetor_total = []

    for i in range(pop):    #Primeira metada = BAIXO, segunda metade = ALTO
        if i>=0 and i <=(((pop/8)*1)-1): #(AAA)
            x2=random.randint(2451,5000)  #o range para o lote de reposição
            x3=random.randint(46,95)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(45,89)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
#print ('Todos os Xs: ', (x1,x2,x3,x4),' com o respectivos construtor de exp: ',
vetor_populacao_identidade[i])
#print ('Desconstruindo fica como: ', divisor_de_expressao(vetor_populacao_identidade[i]))
if i>=((pop/8)*1) and i <=(((pop/8)*2)-1): #(BAA)
x2=random.randint(100,2450)  #o range para o lote de reposição
x3=random.randint(46,95)  #o range para o nível de segurançca
x4=1800+(300*random.randint(45,89)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
        if i>=((pop/8)*2) and i <=(((pop/8)*3)-1): #(ABA)
            x2=random.randint(2451,5000)  #o range para o lote de reposição
            x3=random.randint(5,45)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(45,89)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
        if i>=((pop/8)*3) and i <=(((pop/8)*4)-1): #(AAB)
            x2=random.randint(2451,5000)  #o range para o lote de reposição
```

```
            x3=random.randint(46,95)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(0,44)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
        if i>=((pop/8)*4) and i <=(((pop/8)*5)-1): #(BBA)
            x2=random.randint(100,2450)  #o range para o lote de reposição
            x3=random.randint(5,45)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(45,89)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
        if i>=((pop/8)*5) and i <=(((pop/8)*6)-1): #(BAB)
            x2=random.randint(100,2450)  #o range para o lote de reposição
            x3=random.randint(46,95)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(0,44)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
        if i>=((pop/8)*6) and i <=(((pop/8)*7)-1): #(ABB)
            x2=random.randint(2451,5000)  #o range para o lote de reposição
            x3=random.randint(5,45)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(0,44)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]
        if i>=((pop/8)*7) and i <=(((pop/8)*8)-1): #(BBB)
            x2=random.randint(100,2450)  #o range para o lote de reposição
            x3=random.randint(5,45)  #o range para o nível de segurançca
            x4=1800+(300*random.randint(0,44)) #o range para o tempo de verificação
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4)
            db_variaveis[i]=[x2,x3,x4]

    #print ("\n Vetor do espaço solução inicial está abaixo com
",len(vetor_populacao_identidade)," elementos: \n")
    #print (vetor_populacao_identidade[:])

    rotulos = ['Lot_rep','Niv_seg','T_rev'] #Rótulos para a lista dodb_variaveis
    df = pd.DataFrame.from_records(db_variaveis, columns=rotulos)
    print (df)

    x_df=df[['Lot_rep','Niv_seg','T_rev']]
    xDummies_df=pd.get_dummies(x_df)
    x = xDummies_df.values

    a=0
    b=0
    c=0
    replicacao = 10 #Quantidade de replicações para diminuir a variabilidade dos resultados
    aux = [0]*len(vetor_populacao_identidade)
    aux2 = 0
    aa = 0

    for i in range (replicacao):     #Somente para gerar o programa serial
```

```
        print ('Estamos na replicação: ', i+1)
        x1 += i
        for j in range (len(vetor_populacao_identidade)):
            aa,x2,x3,x4 = divisor_de_expressao(vetor_populacao_identidade[j])

            vetor_populacao_identidade[j] = construtor_de_expressao(x1,x2,x3,x4)
        p=Pool()
        aux=p.map(chama_o_programa,vetor_populacao_identidade)
        p.close()
        p.join()
        for j in range (len(vetor_populacao_identidade)):
            aux2 = vetor_populacao_resultado[j]
            vetor_populacao_resultado[j] = aux2 + aux[j]

    for j in range (len(vetor_populacao_identidade)):
            aux2 = vetor_populacao_resultado[j]
            vetor_populacao_resultado[j] = round ((aux2/replicacao),4)
            #print ('Vetor população resultado: ', vetor_populacao_resultado[j])
            result_y[j]=math.floor(vetor_populacao_resultado[j]/30000) #Para criar classes para o
ML
            #print ('Vetor result_y:', result_y[j])
    y=np.array(result_y) #Transforma o vetor de tipo normal em do tipo NumPy, com os
resultados em CLASSES

    bubble_sort_inv(vetor_populacao_resultado,vetor_populacao_identidade) #Ordena o vetor
para facilitar a seleção dos mais aptos
    print ('Da população inicial temos os 10 melhores valores iniciais:
',vetor_populacao_resultado[:10])

    #A GERAÇÃO INICIAL DA POPULAÇÃO TERMINA AQUI
    #COMEÇA AGORA A FASE ITERATIVA DO ALGORITMO GRASP

    #Fase CONSTRUTIVA do GRASP

    temporario_vetor = [0]*125
    temporario_resultado = [0]*125
    lista_melhores_vetor = [0]*125
    lista_melhores_resultado = [0]*125
    cont = 0
    cont_iter = 0

    #11.2.  Decision Trees Regressor (DTR)
    nome = "Decision Trees Regressor (DTR)"
    from sklearn import tree
    modelo = tree.DecisionTreeRegressor()
    modelo.fit(x, y)

    while ((math.floor( time.time()-t1))<=10800):
        for hh in range (10):
            alfa = (0.8*len(vetor_populacao_identidade))-1
```

```
        a=random.randint(0,alfa) # De 0 a 287 representam os 288 elementos que representam
80%
        b=random.randint(0,alfa) # do espaço solução original, um ALFA para a Lista Restrita
de
        c=random.randint(0,alfa) # Candidatos igual a 80%

        if a==b:                    # Só para garantir que os números serão diferentes
           while (a==b):            # assim os números não se repetirão
              b=random.randint(0,alfa)
        if a==c:
           while (a==c):
              c=random.randint(0,alfa)
        if b==c:
           while (b==c):
              c=random.randint(0,alfa)

        a1, sp1, x2, x3 = divisor_de_expressao(vetor_populacao_identidade[a]) # A Solução
Parcial é criada aqui,
        a2, y1, sp2, y3 = divisor_de_expressao(vetor_populacao_identidade[b]) # formada por
sp1, sp2 e sp3
        a3, z1, z2, sp3 = divisor_de_expressao(vetor_populacao_identidade[c])

        # Início da fase de BUSCA do GRASP

        #Definição da magnitude dos vizinhos em torno da PRIMEIRA variável
         # de maneira tal que seja como: q1|q2|sp1|q4|q5
        if (sp1-100)==0: # Define os valores de vizinhos para a variável  LOTE DE
REPOSIÇÃO
           q1=sp1
           q2=sp1
        if (sp1-100)==1:
           q1=sp1-1
           q2=sp1-1
        if (sp1-100)>=2:
           q1=sp1 - min (int((sp1-100)/2),15)
           q2=sp1 - min (int((sp1-100)/2),7)

        if (sp1-5000)==0:
           q4=sp1
           q5=sp1
        if (sp1-5000)==-1:
           q4=sp1+1
           q5=sp1+1
        if (sp1-5000)<=2:
           q4=sp1 + min (int((sp1-100)/2),7)
           q5=sp1 + min (int((sp1-100)/2),15)

        #Definição da magnitude dos vizinhos em torno da SEGUNDA variável
        # de maneira tal que seja como: w1|w2|sp2|w4|w5
```

```
        if (sp2-5)==0: # Define os valores de vizinhos para a variável NÍVEL DE SEGURANÇA
            w1=sp2
            w2=sp2
        if (sp2-5)==1:
            w1=sp2-1
            w2=sp2-1
        if (sp2-5)>=2:
            w1=sp2-2
            w2=sp2-1

        if (sp2-95)==0:
            w4=sp2
            w5=sp2
        if (sp2-95)==-1:
            w4=sp2+1
            w5=sp2+1
        if (sp2-95)<=2:
            w4=sp2+1
            w5=sp2+2

        #Definição da magnitude dos vizinhos em torno da TERCEIRA variável
        # de maneira tal que seja como: e1|e2|sp3|e4|e5
        if (sp3-1800)==0: # Define os valores de vizinhos para a variável Tempo
            e1=sp3
            e2=sp3
        if (sp3-1800)==300:
            e1=sp3-300
            e2=sp3-300
        if (sp3-1800)==600:
            e1=sp3-600
            e2=sp3-300
        if (sp3-1800)==900:
            e1=sp3-900
            e2=sp3-600
        if (sp3-1800)>=1200:
            e1=sp3-1200
            e2=sp3-600

        if (sp3-28500)==0:
            e4=sp3
            e5=sp3
        if (sp3-28500)==-300:
            e4=sp3+300
            e5=sp3+300
        if (sp3-28500)==-600:
            e4=sp3+300
            e5=sp3+600
        if (sp3-28500)==-900:
            e4=sp3+600
```

```
            e5=sp3+900
        if (sp3-28500)<=-1200:
            e4=sp3+600
            e5=sp3+1200

        Lot = [q1,q2,sp1,q4,q5]
        Seg = [w1,w2,sp2,w4,w5]
        Tem = [e1,e2,sp3,e4,e5]

        marcador=0

        for i in range (len(Lot)):
            for j in range (len(Seg)):
                for k in range (len(Tem)):

temporario_vetor[marcador]=construtor_de_expressao(x1,Lot[i],Seg[j],Tem[k])
                    temporario_vetor_2[marcador]=[Lot[i],Seg[j],Tem[k]]
                    marcador+=1
        temporario_result_2 = modelo.predict(temporario_vetor_2[:125])
        bubble_sort_inv(temporario_result_2,temporario_vetor)

        for qq in range (40):
            temporario_result_total.append(temporario_result_2[qq])
            temporario_vetor_total.append(temporario_vetor[qq])

    bubble_sort_inv(temporario_result_total,temporario_vetor_total)

    x1 = SEMENTE #como o programa usou o x1 antes, tem atribuir o valor anteriormente
dado
    temporario_resultado = [0]*len(temporario_resultado) #limpa o vetor de resultados antes
de voltar a usar
    for i in range (replicacao):     #Somente para gerar o programa serial
        x1 += i
        for j in range (len(temporario_vetor_total[:40])):
            aa,x2,x3,x4 = divisor_de_expressao(temporario_vetor_total[j])
            temporario_vetor_total[j] = construtor_de_expressao(x1,x2,x3,x4)
        p=Pool()
        aux[:40]=p.map(chama_o_programa,temporario_vetor_total[:40])
        p.close()
        p.join()
        for j in range (len(temporario_vetor_total[:40])):
            aux2 = temporario_resultado[j]
            temporario_resultado[j] = aux2 + aux[j]

    for j in range (len(temporario_vetor_total[:40])):
            aux2 = temporario_resultado[j]
            temporario_resultado[j] = round ((aux2/replicacao),4)
    bubble_sort_inv(temporario_resultado[:40],temporario_vetor_total[:40])
```

```
    for i in range (len(temporario_resultado[:40])):
       temporario_vetor_2[i]=[divisor_de_expressao_3(temporario_vetor_total[i])]
       x=np.concatenate([x,temporario_vetor_2[i]], axis=0)

    for i in range (len(temporario_resultado[:40])):    #Depois do POOL dar o APPEND no vetor Y
       y=np.append(y,math.floor(temporario_resultado[i]/30000)) #só numeros inteiros são passados

    modelo.fit(x, y)
    lista_melhores_vetor[cont]=temporario_vetor_total[0]
    lista_melhores_resultado[cont]=temporario_resultado[0]

    antes=lista_melhores_resultado[0]
    bubble_sort_inv(lista_melhores_resultado,lista_melhores_vetor)
    depois = lista_melhores_resultado[0]
    if (antes-depois)==0:
       cont_iter+=1
    else:
       cont_iter=0

    cont+=1
    print ("Contador de iterações igual a: ", cont-1)
    print("Tempo de processamento SERIAL igual a: ",math.floor( time.time()-t1), " segundos, até o momento")
    print ("\n Estamos à ",cont_iter," populações sem melhora na melhor solução \n")
```

Apêndice VIII – Código integrando GRASP, AM e paralelismo ao OE3

```
import time
from multiprocessing import Pool
import math
import itertools
import random
import simpy
import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt
from numpy.random import RandomState

def bubble_sort_inv(seq,res):
    changed = True
    while changed:
        changed = False
        for i in range(len(seq) - 1):
            if seq[i] < seq[i+1]:
                seq[i], seq[i+1] = seq[i+1], seq[i]
                res[i], res[i+1] = res[i+1], res[i]
                changed = True
    return None

def construtor_de_expressao(a,b,c,d,e,f,g,h):
    return
(a*10000000000+b*1000000000+c*10000000+d*1000000+e*10000+f*1000+g*10+h)

def divisor_de_expressao(n):
    a=(math.floor(n/10**10))
    b=(math.floor(n/10**9)-a*10)
    c=(math.floor(n/10**7)-(math.floor(n/10**9)*100))
    d=(math.floor(n/10**6)-(math.floor(n/10**7)*10))
    e=(math.floor(n/10**4)-(math.floor(n/10**6)*100))
    f=(math.floor(n/10**3)-(math.floor(n/10**4)*10))
    g=(math.floor(n/10**1)-(math.floor(n/10**3)*100))
    h=n%10
    return (a,b,c,d,e,f,g,h)

def divisor_de_expressao_3(n):
    a=(math.floor(n/10**10))
    b=(math.floor(n/10**9)-a*10)
    c=(math.floor(n/10**7)-(math.floor(n/10**9)*100))
    d=(math.floor(n/10**6)-(math.floor(n/10**7)*10))
    e=(math.floor(n/10**4)-(math.floor(n/10**6)*100))
```

```
    f=(math.floor(n/10**3)-(math.floor(n/10**4)*10))
    g=(math.floor(n/10**1)-(math.floor(n/10**3)*100))
    h=n%10
    return (b,c,d,e,f,g,h)

def chama_o_programa (ww):

RNG_SEED,capacidade_M1,capacidade_M2,capacidade_M3,capacidade_M4,capacidade_M
5,capacidade_M6,capacidade_M7=divisor_de_expressao(ww)
    ARR_RATE = 0.4
    taxa_M1 = 20.000
    taxa_M3 = 30.000
    taxa_M5 = 12.000
    taxa_M7 = 15.000

    def gerador_produto(env, arr_stream, arr_rate, atraso_inicial=0,
                        tempo_parada=simpy.core.Infinity, prng=RandomState(0)):

        produtos_criados = 0

        yield env.timeout(atraso_inicial)

        while env.now < tempo_parada:

            iat = prng.exponential(1.0 / arr_rate)

            tempo_M1 = prng.exponential(taxa_M1)
            tempo_M2 = 0.00001
            tempo_M3 = prng.exponential(taxa_M3)
            tempo_M4 = 0.00001
            tempo_M5 = prng.exponential(taxa_M5)
            tempo_M6 = 0.00001
            tempo_M7 = prng.exponential(taxa_M7)

            produtos_criados += 1
            obp = produto_fluxo(env, 'Produto{}'.format(produtos_criados),
                                tempo_M1=tempo_M1, tempo_M2=tempo_M2, tempo_M3=tempo_M3,
tempo_M4=tempo_M4, tempo_M5=tempo_M6, tempo_M6=tempo_M6,
tempo_M7=tempo_M7)

            env.process(obp)

            yield env.timeout(iat)

    def produto_fluxo(env, name, tempo_M1, tempo_M2, tempo_M3, tempo_M4, tempo_M5,
tempo_M6, tempo_M7):

        requerer_recurso = env.now
        requerer_recurso1 = M1_unidade.request()
        yield requerer_recurso1
```

```
    yield env.timeout(tempo_M1) # Stay in M1 bed

    requerer_recurso = env.now
    requerer_recurso2 = M2_unidade.request()
    yield requerer_recurso2

    M1_unidade.release(requerer_recurso1)

    yield env.timeout(tempo_M2)
    requerer_recurso = env.now
    requerer_recurso3 = M3_unidade.request()
    yield requerer_recurso3

    M2_unidade.release(requerer_recurso2)

    yield env.timeout(tempo_M3) # Stay in M2 bed
    requerer_recurso = env.now
    requerer_recurso4 = M4_unidade.request()
    yield requerer_recurso4

    M3_unidade.release(requerer_recurso3)

    yield env.timeout(tempo_M4) # Stay in M2 bed
    requerer_recurso = env.now
    requerer_recurso5 = M5_unidade.request()
    yield requerer_recurso5

    M4_unidade.release(requerer_recurso4)

    yield env.timeout(tempo_M5) # Stay in M2 bed
    requerer_recurso = env.now
    requerer_recurso6 = M6_unidade.request()
    yield requerer_recurso6

    M5_unidade.release(requerer_recurso5)

    yield env.timeout(tempo_M6) # Stay in M2 bed
    requerer_recurso = env.now
    requerer_recurso7 = M7_unidade.request()
    yield requerer_recurso7

    M6_unidade.release(requerer_recurso6)

    yield env.timeout(tempo_M7)
    M7_unidade.release(requerer_recurso7)
    if env.now>14400: # 10 dias de warmup
       total_que_saiu_do_sistemas.put(1)


env = simpy.Environment()
```

```
    total_que_saiu_do_sistemas = simpy.Container(env, capacity=99999999999, init=0)

    prng = RandomState(RNG_SEED)

    # Declare Resources to model all units
    M1_unidade = simpy.Resource(env, capacidade_M1)
    M2_unidade = simpy.Resource(env, capacidade_M2)
    M3_unidade = simpy.Resource(env, capacidade_M3)
    M4_unidade = simpy.Resource(env, capacidade_M4)
    M5_unidade = simpy.Resource(env, capacidade_M5)
    M6_unidade = simpy.Resource(env, capacidade_M6)
    M7_unidade = simpy.Resource(env, capacidade_M7)

    # Run the simulation for a while. Let's shut arrivals off after 50 time units.
    runtime = 57600
    stop_arrivals = 57555
    env.process(gerador_produto(env, "Type1", ARR_RATE, 0, stop_arrivals, prng))
    env.run(until=runtime)
    return ((200*total_que_saiu_do_sistemas.level)-
(25000*(capacidade_M1+capacidade_M3+capacidade_M5+capacidade_M7))-
(10000*(capacidade_M2+capacidade_M4+capacidade_M6)))

if __name__=="__main__":
    pop = 600 # TAMANHO DA POPULAÇÃO INICIAL, QUE INFLUENCIA OUTROS
PARÂMETROS DA SIMULAÇÃO

    vetor_populacao_identidade = [0]*pop
    vetor_populacao_resultado = [0]*pop
    db_variaveis = [0]*pop
    result = [0]*pop
    result_y = [0]*pop
    SEMENTE = 42
    x1 = SEMENTE #semente randômica
    t1= time.time()

    temporario_vetor = [0]*126  #vetor para guardar os valores temporários das variaveis
    temporario_vetor_2 = [0]*126 #guarda os valores para depois inserir na data-base
    temporario_result = [0]*126 #vetor para guardar os valores temporários das FOs
    temporario_result_2 = [0]*126
    temporario_result_total = []
    temporario_vetor_total = []

    for i in range(pop):    #Primeira metada = BAIXO, segunda metade = ALTO
        if i>=0 and i <=(((pop/8)*1)-1): #(AAA)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
```

```
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
            #print ('Todos os Xs: ', (x1,x2,x3,x4),' com o respectivos construtor de exp: ', vetor_populacao_identidade[i])
            #print ('Desconstruindo fica como: ', divisor_de_expressao(vetor_populacao_identidade[i]))
        if i>=((pop/8)*1) and i <=(((pop/8)*2)-1): #(BAA)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
        if i>=((pop/8)*2) and i <=(((pop/8)*3)-1): #(ABA)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
        if i>=((pop/8)*3) and i <=(((pop/8)*4)-1): #(AAB)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
        if i>=((pop/8)*4) and i <=(((pop/8)*5)-1): #(BBA)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
        if i>=((pop/8)*5) and i <=(((pop/8)*6)-1): #(BAB)
            x2=random.randint(1,3)  #o range para o lote de reposição
```

```
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
        if i>=((pop/8)*6) and i <=(((pop/8)*7)-1): #(ABB)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]
        if i>=((pop/8)*7) and i <=(((pop/8)*8)-1): #(BBB)
            x2=random.randint(1,3)  #o range para o lote de reposição
            x3=random.randint(1,10)  #o range para o nível de segurançca
            x4=random.randint(1,3) #o range para o tempo de verificação
            x5=random.randint(1,10)
            x6=random.randint(1,3)
            x7=random.randint(1,10)
            x8=random.randint(1,3)
            vetor_populacao_identidade[i]=construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
            db_variaveis[i]=[x2,x3,x4,x5,x6,x7,x8]

    rotulos = ['Cap_M1','Cap_M2','Cap_M3','Cap_M4','Cap_M5','Cap_M6','Cap_M7'] #Rótulos
para a lista dodb_variaveis
    df = pd.DataFrame.from_records(db_variaveis, columns=rotulos)
    print (df)

    x_df=df[['Cap_M1','Cap_M2','Cap_M3','Cap_M4','Cap_M5','Cap_M6','Cap_M7']]
    xDummies_df=pd.get_dummies(x_df)
    x = xDummies_df.values

    a=0
    b=0
    c=0
    replicacao = 10
    aux = [0]*len(vetor_populacao_identidade)
    aux2 = 0
    aa = 0

    for i in range (replicacao):     #Somente para gerar o programa serial
        print ('Estamos na replicação: ', i+1)
        x1 += i
        for j in range (len(vetor_populacao_identidade)):
```

```
            aa,x2,x3,x4,x5,x6,x7,x8 = divisor_de_expressao(vetor_populacao_identidade[j])

            vetor_populacao_identidade[j] = construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
        p=Pool()
        aux=p.map(chama_o_programa,vetor_populacao_identidade)
        p.close()
        p.join()
        for j in range (len(vetor_populacao_identidade)):
            aux2 = vetor_populacao_resultado[j]
            vetor_populacao_resultado[j] = aux2 + aux[j]

    for j in range (len(vetor_populacao_identidade)):
            aux2 = vetor_populacao_resultado[j]
            vetor_populacao_resultado[j] = round ((aux2/replicacao),4)
            #print ('Vetor população resultado: ', vetor_populacao_resultado[j])
            result_y[j]=math.floor(vetor_populacao_resultado[j]/30) #Para criar classes para o ML
            #print ('Vetor result_y:', result_y[j])
    print("Tempo de processamento PARALELO somente para a população igual a: ",math.floor( time.time()-t1), " segundos, até o momento")
    #mode=int(raw_input('Input:'))
    y=np.array(result_y) #Transforma o vetor de tipo normal em do tipo NumPy, com os resultados em CLASSES

    bubble_sort_inv(vetor_populacao_resultado,vetor_populacao_identidade) #Ordena o vetor para facilitar a seleção dos mais aptos
    print ('Primeiros 10 resultados para a população inicial: ',vetor_populacao_resultado[:10])

    #A GERAÇÃO INICIAL DA POPULAÇÃO TERMINA AQUI
    #COMEÇA AGORA A FASE ITERATIVA DO ALGORITMO GRASP

    #Fase CONSTRUTIVA do GRASP

    temporario_vetor = [0]*125
    temporario_resultado = [0]*125
    lista_melhores_vetor = [0]*125
    lista_melhores_resultado = [0]*125
    cont = 0
    cont_iter = 0

    #11.2.  Decision Trees Regressor (DTR)
    nome = "Decision Trees Regressor (DTR)"
    from sklearn import tree
    modelo = tree.DecisionTreeRegressor()
    modelo.fit(x, y)

    posicao = []
    while ((math.floor( time.time()-t1))<=10800000):
        for hh in range (10):    #***####### PARÂMETRO DA ML TROCADO AQUI
            alfa = (0.8*len(vetor_populacao_identidade))-1
            posicao = random.sample(range(0,int(alfa)), 7)
```

```
        # De 0 a 287 representam os 288 elementos que representam 80%
        # do espaço solução original, um ALFA para a Lista Restrita de
        # Candidatos igual a 80%

        a1,sp1,a2,a3,a4,a5,a6,a7 =
divisor_de_expressao(vetor_populacao_identidade[posicao[0]]) # A Solução Parcial é criada
aqui,
        a8,a9,sp2,a10,a11,a12,a13,a14 =
divisor_de_expressao(vetor_populacao_identidade[posicao[1]]) # formada por sp1, sp2 e sp3
        a15,a16,a17,sp3,a18,a19,a20,a21 =
divisor_de_expressao(vetor_populacao_identidade[posicao[2]])
        a22,a23,a24,a25,sp4,a26,a27,a28 =
divisor_de_expressao(vetor_populacao_identidade[posicao[3]])
        a29,a30,a31,a32,a33,sp5,a34,a35 =
divisor_de_expressao(vetor_populacao_identidade[posicao[4]])
        a36,a37,a38,a39,a40,a41,sp6,a42 =
divisor_de_expressao(vetor_populacao_identidade[posicao[5]])
        a43,a44,a45,a46,a47,a48,a49,sp7 =
divisor_de_expressao(vetor_populacao_identidade[posicao[6]])

        # Início da fase de BUSCA do GRASP

        #Definição da magnitude dos vizinhos em torno da PRIMEIRA variável
        # de maneira tal que seja como: q1|q2|sp1|q4|q5
        if (sp2==1 or sp2==2 or sp2==3):
           Lot = [1,2,3,4,5]
        if (sp2==4 or sp2==5 or sp2==6 or sp2==7):
           Lot = [sp2-2,sp2-1,sp2,sp2+1,sp2+2]
        if (sp2==8 or sp2==9 or sp2==10):
           Lot = [6,7,8,9,10]

        #Definição da magnitude dos vizinhos em torno da SEGUNDA variável
        # de maneira tal que seja como: w1|w2|sp2|w4|w5
        if (sp4==1 or sp4==2 or sp4==3):
           Seg = [1,2,3,4,5]
        if (sp4==4 or sp4==5 or sp4==6 or sp4==7):
           Seg = [sp4-2,sp4-1,sp4,sp4+1,sp4+2]
        if (sp4==8 or sp4==9 or sp4==10):
           Seg = [6,7,8,9,10]

        #Definição da magnitude dos vizinhos em torno da TERCEIRA variável
        # de maneira tal que seja como: e1|e2|sp3|e4|e5
        if (sp6==1 or sp6==2 or sp6==3):
           Tem = [1,2,3,4,5]
        if (sp6==4 or sp6==5 or sp6==6 or sp6==7):
           Tem = [sp6-2,sp6-1,sp6,sp6+1,sp6+2]
        if (sp6==8 or sp6==9 or sp6==10):
           Tem = [6,7,8,9,10]

        marcador=0
```

```
        #x1 = SEMENTE

        for i in range (len(Lot)):
            for j in range (len(Seg)):
                for k in range (len(Tem)):
                    #print (" Busca local antes: ",temporario_vetor[marcador])

temporario_vetor[marcador]=construtor_de_expressao(x1,sp1,Lot[i],sp3,Seg[j],sp5,Tem[k],s
p7)
                    temporario_vetor_2[marcador]=[sp1,Lot[i],sp3,Seg[j],sp5,Tem[k],sp7]

                    marcador+=1
        temporario_result_2 = modelo.predict(temporario_vetor_2[:125])

        bubble_sort_inv(temporario_result_2,temporario_vetor)

        for qq in range (40):
            temporario_result_total.append(temporario_result_2[qq])
            temporario_vetor_total.append(temporario_vetor[qq])


    bubble_sort_inv(temporario_result_total,temporario_vetor_total)

    x1 = SEMENTE #como o programa usou o x1 antes, tem atribuir o valor anteriormente
dado
    temporario_resultado = [0]*len(temporario_resultado) #limpa o vetor de resultados antes
de voltar a usar

    for i in range (replicacao):     #Somente para gerar o programa serial
        print ('Estamos na replicação: ', i+1)
        x1 += i
        for j in range (len(temporario_vetor_total[:40])):
            aa,x2,x3,x4,x5,x6,x7,x8 = divisor_de_expressao(temporario_vetor_total[j])

            temporario_vetor_total[j] = construtor_de_expressao(x1,x2,x3,x4,x5,x6,x7,x8)
        p=Pool()
        aux[:40]=p.map(chama_o_programa,temporario_vetor_total[:40])
        p.close()
        p.join()
        for j in range (len(temporario_vetor_total[:40])):
            aux2 = temporario_resultado[j]
            temporario_resultado[j] = aux2 + aux[j]

    for j in range (len(temporario_vetor_total[:40])):
            aux2 = temporario_resultado[j]
            temporario_resultado[j] = round ((aux2/replicacao),4)
    print("Tempo de processamento PARALELO somente para a população igual a:
",math.floor( time.time()-t1), " segundos, até o momento")
    #mode=int(raw_input('Input:'))
    '''
```

```
    for i in range (len(temporario_resultado[:125])):    #Somente para gerar o programa serial
        a=0
        for j in range (replicacao):
            x1,x2,x3,x4 = divisor_de_expressao(temporario_vetor[i])
            x1 += j
            temporario_vetor[i] = construtor_de_expressao(x1,x2,x3,x4)
            b=chama_o_programa(temporario_vetor[i]) #Semente rand, tamanho do lote, nível de segurança, periodo de revisão
            a+=b
        temporario_resultado[i] = round((a/replicacao),4)
    '''

    bubble_sort_inv(temporario_resultado[:40],temporario_vetor_total[:40])

    for i in range (len(temporario_resultado[:40])):
        temporario_vetor_2[i]=[divisor_de_expressao_3(temporario_vetor_total[i])]
        x=np.concatenate([x,temporario_vetor_2[i]], axis=0)

    for i in range (len(temporario_resultado[:40])):    #Depois do POOL dar o APPEND no vetor Y
        y=np.append(y,math.floor(temporario_resultado[i]/30)) #só numeros inteiros são passados

    modelo.fit(x, y)
    lista_melhores_vetor[cont]=temporario_vetor_total[0]
    lista_melhores_resultado[cont]=temporario_resultado[0]
    print ('MELHOR RESULTADO DA RODADA: ', temporario_resultado[:5])

    antes=lista_melhores_resultado[0]
    bubble_sort_inv(lista_melhores_resultado,lista_melhores_vetor)
    depois = lista_melhores_resultado[0]
    if (antes-depois)==0:
        cont_iter+=1
    else:
        cont_iter=0

    print (" Lista dos melhores VALORES: ")
    print (lista_melhores_vetor)
    print (" Lista dos melhores RESULTADOS: ")
    print (lista_melhores_resultado)

    cont+=1
    print ("Contador de iterações igual a: ", cont-1)
    print("Tempo de processamento SERIAL igual a: ",math.floor( time.time()-t1), " segundos, até o momento")
    print ("\n Estamos à ",cont_iter," populações sem melhora na melhor solução \n")
```

APÊNDICE IX - Artigos publicados e submetidos

Neste apêndice são apresentados os artigos publicados e que foram submetidos para avaliação.

Artigos publicados:

- SOUSA JR, W. T. de, MONTEVECHI, J. A. B., MIRANDA, R. C. de. Discrete simulation-based optimization methods for Industrial Engineering problems: A systematic literature review. **Computers & Industrial Engineering**, v. 128, n. January, p. 526–540, 2019.
- SOUSA JR, W. T. de, MONTEVECHI, J. A. B., MIRANDA, R. C. de., ROCHA, F., VILELA, F. F. Economic lot-size using machine learning, parallelism, metaheuristic and simulation. **International Journal of Simulation Modelling**.
- SOUSA JR, W. T. de, MIRANDA, R. C. de, MONTEVECHI, J. A. B. Aprendizado de máquina aplicado à previsão de respostas em problemas de simulação a eventos discretos. In: L Simpósio Brasileiro de Pesquisa Operacional, **Anais…** Rio de Janeiro, 2018.
- SOUSA JR, W. T. de, PEREIRA, R. B. D., MIRANDA, R. C. de, MONTEVECHI, J. A. B., GOMES, J. H. de F. Otimização via simulação a eventos discretos paralela utilizando design de experimentos e metamodelagem. In: Encontro Nacional de Engenharia de Produção, **Anais...** Maceió, 2018.
- SOUSA JR, W. T. de, MONTEVECHI, J. A. B., MIRANDA, R. C. de. Machine learning techniques applied to improve parallel discrete event simulation optimization. In. III Encontro de Pesquisa e Pós-Graduação em Engenharia de Produção, **Anais...** Florianópolis, 2018. (* O trabalho recebeu o prêmio de melhor trabalho de doutorado em Engenharia de Produção).

Artigo esperando revisão:

- SOUSA JR, W. T. de, MONTEVECHI, J. A. B., MIRANDA, R. C. de., OLIVEIRA, M. L. M. de, CAMPOS, A. T. Machine learning techniques applied to improve parallel discrete event simulation for shop floor optimization. **Expert Systems with Applications.**