

METAMODELAGEM ADAPTATIVA: Uma Aplicação na Indústria Têxtil

João Victor Soares do Amaral

Universidade Federal de Itajubá
Avenida BPS, 1303, Itajubá –MG
joao.amaral@unifei.edu.br

José Arnaldo Barra Montevechi

Universidade Federal de Itajubá
Avenida BPS, 1303, Itajubá –MG
montevechi@unifei.edu.br

Rafael de Carvalho Miranda

Universidade Federal de Itajubá
Avenida BPS, 1303, Itajubá –MG
rafael.miranda@unifei.edu.br

Carlos Henrique dos Santos

Universidade Federal de Alfenas
Rodovia José Aurélio Vilela, 11999, Poços de Caldas – MG
carlos.henrique@unifal-mg.edu.br

RESUMO

A simulação, um pilar da Indústria 4.0, é uma ferramenta poderosa usada para avaliar sistemas sob várias condições, facilitando a análise de desempenho e a tomada de decisões. No entanto, pode ser demorada, especialmente na otimização de modelos complexos. Neste caso, a metamodelagem tem se destacado como uma técnica popular para a otimização via simulação. Este artigo apresenta o AMSO (Adaptive Metamodeling-based Simulation Optimization), um *framework* inovador que busca melhores soluções com menos experimentos. A abordagem combina técnicas de Aprendizado de Máquina e metaheurísticas para identificar as áreas mais promissoras do espaço de solução e explorá-las de forma eficiente. O *framework* foi avaliado em um problema real de alocação de recursos de um modelo de Gêmeo Digital de manufatura. Comparado ao Algoritmo Genético, o AMSO encontrou uma solução estatisticamente igual, mas exigiu 82% menos tempo computacional.

PALAVRAS-CHAVE. Metamodelagem. Otimização via Simulação. Simulação a Eventos Discretos.

Tópicos. SIM - Simulação

ABSTRACT

Simulation, a pillar of Industry 4.0, is a powerful tool used to evaluate systems under various conditions, facilitating performance analysis and decision-making. However, it can be time-consuming, especially when optimizing complex models. In this case, metamodeling has emerged as a popular technique for simulation optimization. This paper presents AMSO (Adaptive Metamodeling-based Simulation Optimization), an innovative *framework* that aims for better solutions with fewer experiments. The approach combines Machine Learning and metaheuristic techniques to identify and efficiently explore the most promising areas of the solution space. The *framework* was evaluated in a real-world resource allocation problem of a manufacturing Digital Twin model. Compared to the Genetic Algorithm, AMSO found a statistically equal solution but required 82% less computational time.

KEYWORDS. Metamodeling. Simulation Optimization. Discrete Event Simulation.

Paper topics. SIM - Simulation

1. Introdução

Segundo Law [2013], a técnica de Simulação a Eventos Discretos (SED) visa representar computacionalmente sistemas reais, estudando o comportamento de variáveis que mudam de estado em horizontes de tempo discretos. A SED reduz o risco na tomada de decisão, permitindo a avaliação de diferentes cenários sem interferência física no sistema de produção [Guimarães *et al.* 2018]. No entanto, muitas organizações não aproveitam os benefícios da simulação devido ao tempo e à complexidade envolvidos nos projetos de simulação [Barlas e Heavey 2016]. Para adaptar-se às exigências da indústria moderna, especialmente os princípios da Indústria 4.0, a simulação está passando por mudanças significativas [Rodric 2017].

A análise de sistemas de produção complexos envolve diversas variáveis de decisão e uma função objetivo, resultando na avaliação de inúmeros cenários. Neste caso, técnicas de otimização via simulação (OvS) são importantes para identificar o melhor cenário [Xu *et al.* 2016]. No entanto, simulações podem ser computacionalmente intensivas, especialmente com modelos complexos, tornando o processo de OvS impraticável quando o tempo de decisão é limitado. Para superar isso, sugere-se elevar o problema a um nível mais alto de abstração, conhecido como metamodelagem [Chabanet *et al.* 2021; Yousefi e Yousefi 2019].

A metamodelagem estabelece uma conexão simplificada entre as entradas e saídas de um modelo de simulação, usada para avaliar soluções durante o processo de otimização [Rojas Gonzalez *et al.* 2020]. Neste contexto, substituir o modelo de simulação pelo metamodelo pode reduzir o tempo de otimização [Parnianifard *et al.* 2019]. Vários estudos propuseram técnicas de metamodelagem mais precisas e eficientes [Moghaddam e Mahlooji 2017].

As abordagens de metamodelagem podem ser categorizadas como *design* experimental fixo ou estratégia incremental [Amaral *et al.* 2022a]. No *design* fixo, um conjunto experimental é escolhido no início do projeto e o metamodelo é treinado apenas nesse conjunto. Na estratégia incremental, devido a um orçamento de simulação limitado, os experimentos são alocados a cada iteração em regiões mais promissoras do espaço de busca. Este trabalho foca na estratégia incremental.

Neste sentido, este trabalho apresenta uma nova abordagem de metamodelagem, chamada *Adaptive Metamodeling-based Simulation Optimization* (AMSO), para resolver problemas de otimização em SED. A estrutura proposta combina técnicas de *design of experiments* (DOE), *machine learning* e metaheurísticas visando alcançar bons resultados na OvS utilizando um *budget* restrito de experimentos. Detalhes dessas técnicas são explicados na próxima seção.

A aplicabilidade e desempenho do AMSO é demonstrado na otimização de um modelo de SED para alocação periódica de recursos em uma fábrica têxtil. Como *benchmark*, este trabalho compara os resultados com aqueles obtidos com a aplicação do Algoritmo Genético (AG). Além disto, será discutido a sensibilidade do método AMSO a um importante parâmetro: a função de aquisição (FA) utilizada para guiar o processo de busca do método. As principais contribuições deste artigo são:

- Apresentar uma nova estrutura de otimização baseada em metamodelagem, chamada AMSO, que combina técnicas de aprendizado de máquina, metaheurísticas e *design* de experimentos para encontrar melhores valores de função objetivo com um *budget* de simulação limitado.
- Demonstrar o desempenho da AMSO aplicando-a a um problema real de OvS, comparando os resultados com método um método de OvS amplamente utilizado: o AG.

O restante deste artigo está dividido em quatro seções adicionais. A Seção 2 apresenta o embasamento teórico, a Seção 3 descreve a estrutura proposta, a Seção 4 detalha sua aplicação em um objeto de estudo, e a Seção 5 conclui o trabalho e propõe direções para pesquisas futuras.

2. Referencial Teórico

Segundo Xu *et al.* [2016], a OvS visa otimizar uma função objetivo sujeita a restrições usando simulação estocástica para avaliação. O processo envolve determinar os valores de entrada ideais para variáveis do sistema simulado e avaliar as soluções por meio de um *loop* iterativo entre o algoritmo de otimização e o modelo de simulação. Apesar dos avanços significativos, o desafio do tempo computacional persiste.

Sousa Junior *et al.* [2019] enfatizam que em sistemas complexos com grande espaço de solução, o tempo computacional necessário para o algoritmo de otimização convergir é longo. Barton [2009] sugere métodos especializados para OvS, como *ranking* e seleção, heurísticas e metaheurísticas, busca aleatória e metamodelagem.

De La Fuente, Erazo e Smith [2019] destacam que a complexidade do sistema impacta diretamente o tempo necessário para otimizar. Recomenda-se o uso de metamodelagem, que cria um modelo representativo do modelo de simulação, capturando a relação entre variáveis de decisão e saídas da simulação, proporcionando uma aproximação mais rápida da função objetivo do que a simulação. Sousa Junior *et al.* [2020] também destacam a eficácia da metamodelagem na redução do tempo de otimização.

Amaral *et al.* [2022a] categorizaram as estratégias de metamodelagem em *design* experimental fixo e *design* incremental. A estratégia fixa envolve tomar um único conjunto de amostras no início do projeto. Já a estratégia incremental começa com um conjunto inicial de dados para treinar e otimizar o metamodelo, adicionando novas amostras em cada iteração. O metamodelo é retreinado com uma base de tamanho crescente conforme as iterações progridem. O novo ponto a ser simulado a cada iteração é definido por meio da otimização de uma função de aquisição (FA), a qual utiliza previsões do metamodelo para sinalizar o valor intrínseco de cada ponto do espaço de busca para o problema, seja como possível ponto de ótimo ou como ponto que traga mais informações e reduza o erro do metamodelo.

Dentre as diversas FAs propostas na literatura, destaca-se a *Balanced Expected Improvement* (BEI) [Montevecchi *et al.* 2024], *Expected Improvement* (EI) [Wang e Ierapetritou 2017], *Surrogate-based Optimization* (SBO) [Fan e Hu 2018], *Probability of Improvement* (PoI) [Yin, Liu e Zheng, 2022], *Upper Bound Confidence* (UBC) e *Lower Bound Confidence* (LBC) [Candelieri, Perego e Archetti 2018]. A formulação da função BEI é definida pela Equação (1) [Montevecchi *et al.* 2024].

$$BEI(\mathbf{x}) = (1 - \lambda)\sigma\phi\left(\frac{f_{min} - \hat{f}(\mathbf{x})}{\sigma}\right) + \lambda\hat{f}(\mathbf{x}) \quad (1)$$

A Equação (1) utiliza várias variáveis, incluindo ϕ como a função de densidade de probabilidade, f_{min} como o menor valor observado no conjunto de dados de treinamento, e $\hat{f}(\mathbf{x})$ e σ como a previsão e o desvio padrão, respectivamente, do metamodelo para o ponto $\mathbf{x}$. O componente de Exploração é descrito pelo primeiro termo da Equação (1), enquanto o segundo termo representa o componente de Exploração. O valor de λ determina se a função de aquisição prioriza a Exploração ou Exploração. O valor de λ é determinado pelo R^2 calculado no passo anterior e o valor sorteado de ρ , onde $\rho \sim Uniform(a, b)$, com a e b sendo os limites inferior e superior do sorteio definidos pelo usuário, respectivamente. A Equação (2) estabelece que um R^2 maior indica uma maior probabilidade de Exploração ($\lambda=1$).

$$\lambda = \begin{cases} 1, & \text{if } \rho < R^2 \\ 0, & \text{Caso contrário} \end{cases} \quad (2)$$

A formulação tradicional da EI também envolve o balanceamento entre regiões promissoras de conterem o ótimo e regiões com alta incerteza de previsão, assim como demonstrado na Equação (3) [Wang e Ierapetritou 2017].

$$E[I(x)] = (f_{min} - \hat{f}(x))\Phi\left(\frac{f_{min} - \hat{f}(x)}{\sigma(x)}\right) + \sigma(x) \phi\left(\frac{f_{min} - \hat{f}(x)}{\sigma(x)}\right) \quad (3)$$

O menor valor observado no conjunto de dados de treinamento é denotado como f_{min} , a previsão do metamodelo para um determinado ponto x é representado por $\hat{f}(x)$, e $\sigma(x)$ é o desvio padrão da previsão. Φ e ϕ referem-se às funções de distribuição cumulativa normal e densidade de probabilidade. Outra função muito utilizada é a PoI, na qual a ideia básica envolve calcular a probabilidade de que uma nova solução candidata melhore o desempenho em relação à melhor solução atualmente conhecida. Para fazer isso, ela utiliza a distribuição de probabilidade estimada para o valor da função objetivo no espaço de busca. Assim como demonstrado na Equação (4), é possível observar que a EI é uma expansão da formulação da PoI [Yin, Liu e Zheng, 2022].

$$PoI(x) = \Phi\left(\frac{f_{min} - \hat{f}(x)}{\sigma(x)}\right) \quad (4)$$

A função SBO concentra esforços em regiões promissoras de conterem o ótimo, não considerando a incerteza de previsão do metamodelo. A SBO é ideal quando o metamodelo apresenta uma boa aproximação em diferentes regiões do espaço de busca, ou seja, um baixo erro associado. Embora seja a mais simples das FA, ela apresenta excelentes resultados em diversas áreas [Fan e Hu, 2018]. A SBO pode ser formulado seguindo a Equação 5.

$$PB(x) = \hat{f}(x) \quad (5)$$

Por fim, a *Upper/Lower Confidence Bound* (UCB/LCB) é baseada na construção de intervalos de confiança em torno das previsões feitas por um modelo probabilístico da função objetivo. Esses intervalos permitem selecionar pontos com alta incerteza, facilitando a exploração de regiões desconhecidas do espaço de busca [Candelieri, Perego e Archetti 2018]. Estas FAs podem ser descritas pelas Equações (6) e (7), respectivamente.

$$UBC(x) = \hat{f}(x) + 3\sigma(x) \quad (6)$$

$$LBC(x) = \hat{f}(x) - 3\sigma(x) \quad (7)$$

Outro fator crucial no sucesso do método, tanto no *design* fixo quanto incremental, é o algoritmo de *machine learning* utilizado para construção dos metamodelos. Diversos algoritmos foram desenvolvidos para melhorar a velocidade de execução, precisão e confiabilidade da metamodelagem. Em uma revisão sistemática da literatura por Amaral *et al.* [2022a], Kriging, Regressão Polinomial, Redes Neurais, GBT, *Random Forest*, Função de Base Radial e *Support Vector Machine* foram identificados como os principais metamodelos usados em OvS. Estudos anteriores também concluíram que o algoritmo GBT apresentou o melhor desempenho [Amaral *et al.* 2022b; Amaral *et al.* 2022c]. Baseado nesses achados, este estudo propõe uma abordagem de metamodelagem baseada no algoritmo *Bagging* com GBT como base.

O GBT é um algoritmo de Aprendizado de Máquina (ML) utilizado em várias aplicações, como problemas de regressão e classificação [Friedman 2002]. O conceito subjacente do GBT é o "*boosting*", implementado pela construção iterativa de uma sequência de árvores de decisão. Cada árvore corrige os erros das anteriores, combinando suas previsões para obter o modelo final [Zhang e Haghani 2015].

O algoritmo *Bagging* (*Bootstrap Aggregating*) é uma técnica de *Ensemble* que combina múltiplos modelos de ML (*base-learners*) treinados em conjuntos de dados amostrados aleatoriamente usando a técnica de *bootstrapping* [Breiman 1996]. O conjunto de dados original, D , contém n pontos de dados, e B representa o número de modelos a serem treinados. Cada subconjunto D_i consiste em m pontos de dados amostrados aleatoriamente com substituição de D . Para cada $b = 1, 2, \dots, B$, o modelo B_b é treinado usando o subconjunto D_i e um algoritmo de aprendizado supervisionado, que no nosso caso é o GBT [Hastie *et al.* 2008].

O R^2 *out-of-bag* (OOB) é usado para avaliar o desempenho do modelo *Bagging*, calculado usando as amostras de treinamento não selecionadas para um determinado modelo durante o processo de *bagging*. O algoritmo de *Bagging* usa todos os modelos não amostrados durante o treinamento para prever cada amostra OOB, produzindo um conjunto de previsões para cada amostra OOB, sobre o qual o R^2 é calculado [Schonlau e Zou 2020].

Após o treinamento, as previsões do modelo *Bagging* $\hat{f}(x)$ são determinadas pela agregação das previsões de cada base learner, $\hat{f}^{*b}(x)$. Esse processo gera uma previsão final mais precisa e estável do que previsões feitas por modelos individuais [Hastie *et al.* 2008].

Louk e Tama [2023] sugeriram que usar *Gradient Boosted Trees* (GBT) como base learner para *Bagging* pode proporcionar muitos benefícios. *Bagging* pode reduzir a variância geral do modelo, levando a maior precisão nas previsões. Além disso, a natureza sequencial do GBT permite que modelos subsequentes corrijam erros dos modelos anteriores, minimizando o viés das previsões. *Bagging* também aumenta a robustez do modelo ao mitigar o impacto de *outliers* ou dados desbalanceados no conjunto de dados original. Combinar essas técnicas pode reduzir a variância e o viés do modelo [Ganjisaffar *et al.* 2011].

A seleção de hiperparâmetros é crucial durante o treinamento do metamodelo. Segundo Amaral *et al.* [2022b], os principais hiperparâmetros do GBT incluem o número de estimadores, a taxa de aprendizado e o número mínimo de amostras por folha. Devido à dificuldade de definir esses parâmetros manualmente, técnicas de otimização são essenciais para determinar a configuração ideal. Métodos de otimização incluem AG, otimização por colônia de formigas, busca em grade e otimização Bayesiana. AG é uma técnica baseada em seleção natural, combinando genes para gerar novas soluções [Bergstra *et al.* 2012].

Para otimizar os hiperparâmetros e treinar o GBT, AG utiliza validação cruzada *k-fold*, dividindo os dados de treinamento em *k* partes iguais, usando *k-1* partes para treinamento e a parte restante para teste. O erro associado a cada conjunto de parâmetros é calculado *k* vezes, e a média dos erros é utilizada para encontrar a melhor configuração.

3. Método

Esta seção descreve a abordagem proposta.. A Figura 1 apresenta o *framework* AMSO, que se baseia em quatro técnicas: *Latin Hypercube Design*, algoritmo *Gradient-boosted Trees*, *Bagging ensemble* e Algoritmo Genético.

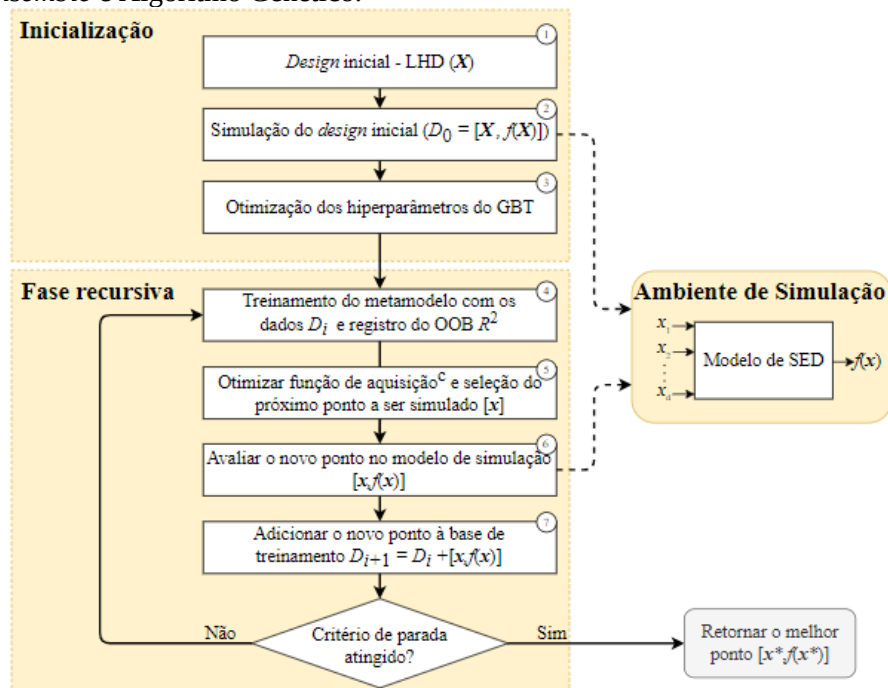


Figura 1: *Framework* proposto.

O *framework* envolve duas fases principais: a fase de inicialização e a fase de *loop*. O primeiro passo da fase de inicialização cria a matriz inicial ($D_0 = [X, f(X)]$) usada para treinar o metamodelo. A técnica *Latin Hypercube Design* (LHD) é empregada para gerar essa matriz, sendo o método de amostragem utilizado em estudos de metamodelagem [Amaral *et al.*, 2022a]. O método LHD divide o intervalo de cada variável em n intervalos disjuntos com igual probabilidade, constrói uma matriz $n \times d$, onde d é o número de variáveis e n é o número de amostras, e seleciona um ponto de cada célula da matriz. Segundo Bharaj *et al.* [2015] e Wang e Ierapetritou [2017], o tamanho da amostra sugerido para a matriz LHD inicial é entre $2d$ a $10d$, onde d é a dimensão do problema. No Passo 2, o arranjo experimental X é avaliado no modelo de simulação, gerando a resposta $f(X)$ para os pontos e criando a base de treinamento inicial para o metamodelo.

No Passo 3, o Algoritmo Genético (AG) otimiza os hiperparâmetros do GBT. Os intervalos de valores possíveis de cada hiperparâmetro são o número de estimadores (1 - 100), a taxa de aprendizado (0,05 - 0,30) e o mínimo de amostras por folha (0,12 - 0,50) [Amaral *et al.*, 2022b].

O Passo 4 marca o início da fase recursiva do AMSO, onde o metamodelo é treinado. O algoritmo Bagging, usando o GBT otimizado como base, é treinado recursivamente com a base de treinamento D_i de tamanho ξ . O algoritmo adiciona um ponto a cada iteração i , $\xi_{i+1} = \xi_i + 1$. Adicionalmente, o R^2 (OOB) do metamodelo é calculado neste passo, que será usado no passo subsequente.

O Passo 5 envolve otimizar a função de aquisição, que é usada para determinar o ponto mais apropriado a ser simulado e incluído no conjunto de dados de treinamento durante a iteração subsequente do algoritmo. O método AMSO permite a utilização de diversas FAs, a depender da preferência do usuário. As FAs implementadas no AMSO são: BEI, EI, PoI, SBO, UCB e LCB. Na seção de resultados será apresentado uma comparação dos resultados obtidos com cada uma destas FAs, demonstrando a sensibilidade do método a este parâmetro. O objetivo desta etapa é identificar o ponto x que maximiza a função de aquisição, o que é realizado empregando AG como motor de busca.

O Passo 6 envolve avaliar a solução ótima encontrada pelo AG no modelo de simulação, resultando no valor de $f(x)$ para este ponto. No Passo 7, o novo ponto é então adicionado ao conjunto de dados de treinamento do metamodelo, $f(x)$. Os Passos 4 a 7 são repetidos até que o critério de parada seja atendido. O número máximo de iterações é o critério de parada neste estudo, e uma vez que esse limite é alcançado, o algoritmo conclui, e a solução x^* com o melhor valor de $f(x^*)$ é retornada como a solução do problema. Outros critérios de parada poderiam ser usados, como o número máximo de iterações sem melhoria ou ajuste do metamodelo; no entanto, considerando que o tempo é uma limitação para os casos, o critério de parada escolhido foi o número máximo de iterações. Este *framework* foi implementado na linguagem *Python* (Versão 3.0).

4. Resultados

4.1. Apresentação do Objeto de Estudo

O objeto de estudo representa uma minifábrica do setor de *fast-fashion*, caracterizada por rápida mudança na demanda, ciclo curto, alta volatilidade e alta competição de preço. O modelo de SED representa os fluxos produtivos, restrições operacionais, variabilidades dos processos, e falhas de equipamentos. O fluxo de produção inclui oito postos de trabalho (Processo B ao I), além de Recepção (A) e Expedição (J). A operação é majoritariamente manual, exceto parte do transporte interno automatizado. A linha conta com operadores multifuncionais divididos em dois grupos de até dez operadores cada: Grupo I (Processos C, D, E, G) e Grupo II (Processos F, H, I, J).

A empresa utiliza previsões semanais de demanda para alocar recursos produtivos e equipe, podendo locar equipamentos e funcionários de outras minifábricas a um custo adicional [Dos Santos *et al.* 2021]. O modelo visa otimizar o lucro da minifábrica por meio da OvS. As

variáveis de decisão são o tamanho da equipe (x_1 e x_2) e a quantidade de equipamentos alocados (x_3 a x_8).

Os limites inferiores e superiores das variáveis são $l = [2 \ 2 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2]^T$ e $u = [10 \ 10 \ 3 \ 10 \ 10 \ 10 \ 10 \ 10]^T$, respectivamente, e o custo unitário é $c = [800 \ 800 \ 500 \ 500 \ 500 \ 300 \ 350 \ 300]^T$. A função lucro é dada pela Equação (8)

$$L(x) = \psi v - \sum_{i=1}^8 x_i c_i \quad (8)$$

Onde $L(x)$ é o lucro, ψ é a produção semanal e v é a margem de contribuição média dos produtos. A função objetivo compreende a maximização da expectativa para a função lucro, $\max f(x) = \mathbb{E}_{\omega}\{\tilde{\phi}[L(x, \omega)]\}$, que pode ser obtida pela média de $L(x)$ das replicações do modelo de simulação ($\tilde{\phi}$), sujeitas a sementes aleatórias ω distintas. Para este modelo foram utilizadas 15 replicações (quantidade mínima de replicações considerando a variabilidade do modelo). O modelo matemático que representa o objeto de estudo é dado pela maximização da função Equação (9), sujeito as restrições (Equação (10) e Equação (11) referentes ao espaço de busca).

$$\max f(x) = \mathbb{E}_{\omega}\{\tilde{\phi}[L(x, \omega)]\} \quad (9)$$

Sujeito a:

$$l_i \leq x_i \leq u_i \quad (10)$$

$$x_i \in \mathbb{Z}, \quad \forall i \in \mathbb{Z} (1, \dots, 8) \quad (11)$$

Vale ressaltar que o valor da FO é obtido apartir da produção encontrada (e consequentemente o valor de $L(x)$ pelo modelo de SED (representado por $\tilde{\phi}$), que compreende diversas restrições operacionais, estocasticidade, interdependências e complexidades não esboçadas no modelo matemático.

O *framework* AMSO proposto foi aplicado para otimizar o problema de OvS com um *budget* limitado de simulações. O critério de parada foi o número máximo de simulações, com 50% destinado a base de treinamento inicial e o restante à fase recursiva. O número máximo de simulações foi definido em 80 experimentos (10 vezes a dimensão do problema). O processo de otimização foi repetido 10 vezes para avaliar robustez e resposta do método.

A fim de avaliar a sensibilidade do método à função de aquisição adotada, a Seção 4.2 apresenta os resultados da aplicação do método considerando as funções BEI, EI, PoI, SBO, UBC e LBC. Em sequência, a Seção 4.3 compara os resultados obtidos com o método AMSO com aqueles obtidos pelo AG.

4.2. Resultados do AMSO

Esta seção apresenta os resultados da aplicação do método AMSO ao objeto de estudo. Inicialmente avaliou-se a sensibilidade do método à função de aquisição utilizada. Nesta comparação foi utilizado o *budget* de simulação de 80 experimentos com 15 replicações cada, sendo o processo de otimização repetido 10 vezes a fim de analisar a robustez do método perante aleatoriedade. A Figura 2 apresenta o resultado da otimização para cada função de aquisição.

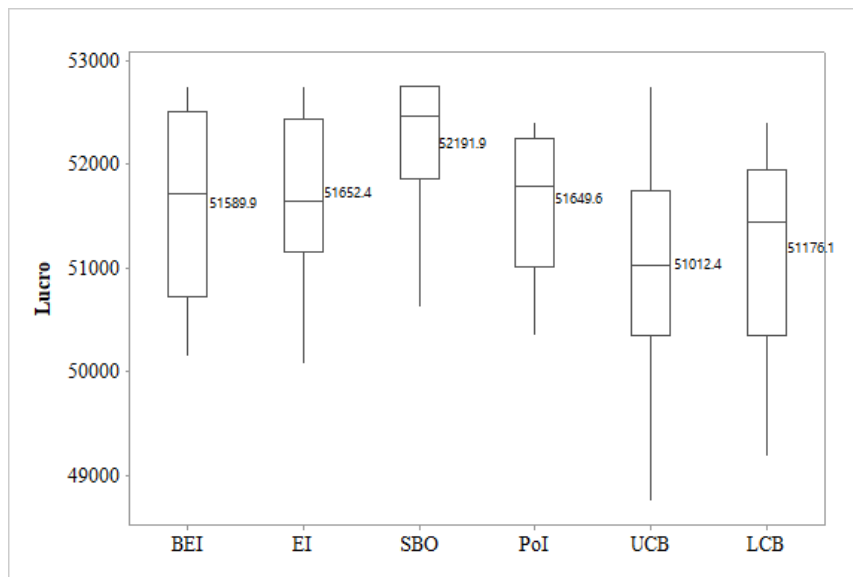


Figura 2: Box-plot dos resultados para cada função de aquisição.

Conforme pode-se observar na Figura 2, embora a função de aquisição SBO tenha apresentado melhor média para este objeto de estudo, não é possível afirmar que existe diferença significativa entre os resultados obtidos para cada função de aquisição (p-value de 0,162 para o teste ANOVA com 5% de significância). Desta forma, pode-se concluir que, para o objeto de estudo avaliado, o método AMSO se mostrou robusto quanto ao tipo de função de aquisição utilizado. Como não houve diferença estatisticamente significativa entre as FAs, optou-se por utilizar aquela com maior mediana, ou seja, a SBO.

A Figura 3 apresenta o gráfico de convergência dos resultados obtidos com o AMSO utilizando a função de aquisição SBO. Em média o método proposto alcançou um lucro de R\$ 52.191,85 e desvio padrão de R\$ 826,89, com resultados variando entre R\$ 50.621,60 e R\$ 52.750,40. A melhor solução encontrada pelo AMSO foi $x^* = [8 \ 2 \ 3 \ 2 \ 2 \ 2 \ 2 \ 2]$ e demandou, em média, um tempo computacional de 1.484 segundos para finalizar a otimização.

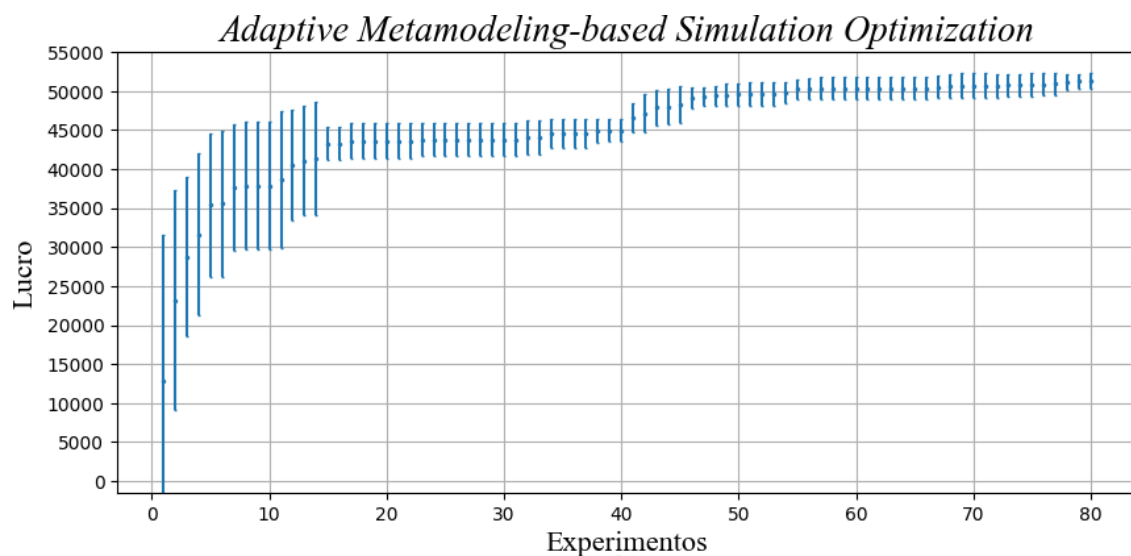


Figura 3: Gráfico de convergência para o método AMSO com a função SBO.

4.3 Resultados do Algoritmo Genético

O segundo *benchmark* empregado foi o AG, implementado a partir do pacote *Python geneticalgorithm2* [Pascal e Solgi 2023] do PyPI. Considerando que o AG não trabalha com um número pequeno de avaliações de funções (experimentos), este teste não o limitou ao mesmo *budget* aplicado ao AMSO. O AG foi definido com um número considerável de gerações para analisar se e quando este método supera os resultados obtidos pelo AMSO. Os parâmetros do AG foram definidos como 20 indivíduos para o tamanho da população e o número máximo de iterações igual a 100. Os demais parâmetros foram configurados conforme o padrão do pacote *Python geneticalgorithm2*. A Figura 4 apresenta a média e o erro do gráfico de convergência do AG (10 repetições).

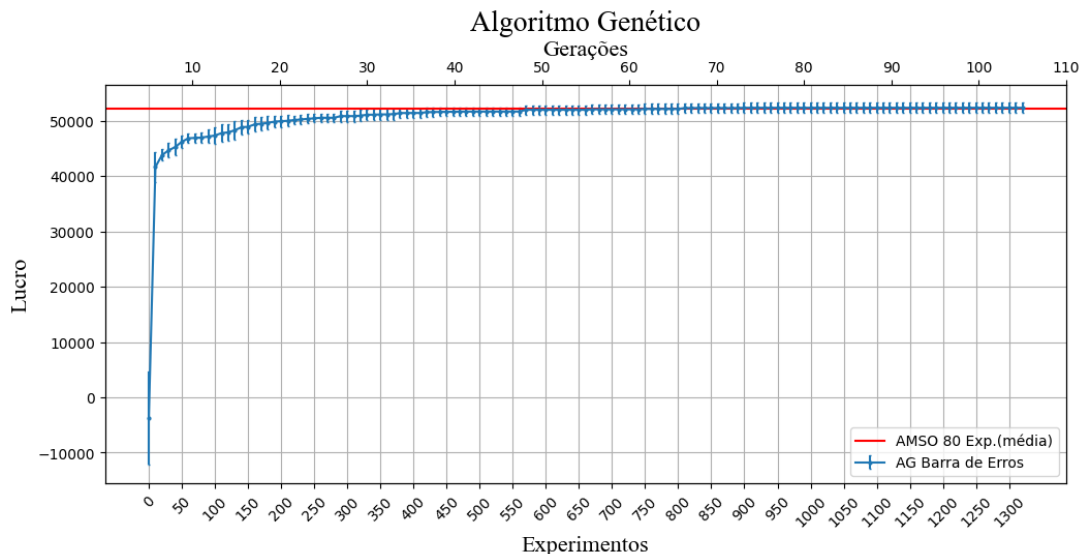


Figura 4: Gráfico de convergência do algoritmo genético.

Após as 100 gerações (mais de 1.300 experimentos), o AG atingiu um lucro médio de R\$ 52.377,62, necessitando em média de 18.653,89 segundos para finalização do processo de otimização. Conforme destacado na Figura 4, os resultados obtidos com o AG foram estatisticamente iguais aos obtidos pelo método AMSO com apenas 80 experimentos (p-value de 0,09 para o teste de Mann-Whitney a 5% de significância), todavia demandando 1257 vezes mais tempo computacional que o AMSO. Ressalta-se que os resultados obtidos com o AMSO se equiparam aos obtidos com o AG, porém demandando consideravelmente menos experimentos (e tempo), o que mostra a eficiência do *framework* proposto para esta aplicação. Vale ressaltar que não é objetivo analisar a superioridade do AMSO sobre o AG, uma vez que ambos pertencem a classes de algoritmos de otimização distintas e possuem aplicações e características particulares.

5. Conclusão e Oportunidades Futuras

O presente estudo introduz o *framework* AMSO, que integra o *Design* de Experimentos, modelo *Bagging*, GBT, otimização de hiperparâmetros e Metaheurísticas (AG) para otimizar modelos SED com um *budget* limitado de experimentos. O método consiste em dez etapas principais, conforme detalhado na Seção 3, e visa treinar um metamodelo usando uma estratégia de amostragem incremental. Esta estratégia seleciona um novo ponto para simular e adicionar ao banco de dados de treinamento do metamodelo em cada iteração. A seleção do novo ponto é baseada na otimização da função de aquisição via AG, que equilibra a exploração e a exploração usando previsões de metamodelos previamente treinados. Como função de aquisição, este trabalho explora as funções EI, SBO, PoI, UCB, LCB e BEI.

Este artigo apresenta um estudo de caso real para exemplificar a aplicabilidade do AMSO. Também aplicou-se o método AG para fins de comparação. O caso compreende um modelo digital de uma planta de manufatura *fast-fashion*, otimizando recursos produtivos para atender à demanda semanal prevista. O objetivo é maximizar o lucro da fábrica, trabalhando com

oito variáveis de decisão discretas. Nos métodos AMSO, o *budget* de simulação foi limitado a 80. No entanto, devido às suas características, o AG foi configurado com um *budget* de simulação maior para avaliar quando ele supera os resultados do AMSO.

Os resultados demonstraram a superioridade do *framework* proposto. Em média, o AMSO encontrou uma solução com um lucro de R\$52.191,85. Além disso, o AG terminou após executar 100 gerações (aproximadamente 1320 experimentos) com um resultado estatisticamente igual ao AMSO com apenas 80 experimentos, que demandou 12,57 vezes menos tempo do que o AG.

O *framework* proposto neste estudo produziu resultados consistentes para um problema típico em SED: alta estocasticidade, interdependência de processos, não-linearidade e potencial descontinuidade da função objetivo. Essas características necessitam de metamodelos com alta capacidade de aprendizado, como os empregados nesta pesquisa. Com base nos resultados apresentados, destaca-se o potencial do *framework* AMSO para resolver problemas de OvS complexos em aplicações em que o tempo de tomada de decisões é uma restrição. Outra vantagem do AMSO é que, ao final da otimização, um de seus resultados é um metamodelo treinado, que o tomador de decisão pode usar para explorar outros cenários potenciais e realizar otimização com novas restrições sem necessitar de execuções de simulação dispendiosas, acelerando o processo de tomada de decisão.

Este artigo utiliza GBT como o *base-learner* do *Bagging*. Assim, uma extensão de pesquisa avalia e compara sistematicamente outros algoritmos de aprendizado de máquina, como *Support Vector Machine*, Redes Neurais Artificiais e *Extreme Gradient-boosting*. Além disto, sugere-se que estudos futuros avaliem e comparem o AMSO para novos problemas SED ou outros problemas de otimização.

Referências

- Amaral, J. V. S. do, Montevechi, J. A. B., Miranda, R. de C., & Junior, W. T. de S. (2022a). Metamodel-based simulation optimization: A systematic literature review. *Simulation Modelling Practice and Theory*, 114, 102403. <https://doi.org/10.1016/j.simpat.2021.102403>
- Amaral, J. V. S. do, Miranda, R. de C., Montevechi, J. A. B., dos Santos, C. H., & da Silva, A. F. (2022b). Data envelopment analysis for algorithm efficiency assessment in metamodel-based simulation optimization. *International Journal of Advanced Manufacturing Technology*, 121(11–12), 7493–7507. <https://doi.org/10.1007/s00170-022-09864-z>
- Amaral, J. V. S. do, Miranda, R. de C., Montevechi, J. A. B., dos Santos, C. H., & Gabriel, G. T. (2022c). Metamodeling-based simulation optimization in manufacturing problems: a comparative study. *The International Journal of Advanced Manufacturing Technology*, 20. <https://doi.org/10.1007/s00170-022-09072-9>
- Barlas, P., & Heavey, C. (2016). Automation of input data to discrete event simulation for manufacturing: A review. *International Journal of Modeling, Simulation, and Scientific Computing*, 7(1). <https://doi.org/10.1142/S1793962316300016>
- Barton, R. R. (2009). Simulation Optimization Using Metamodel. In *Winter Simulation Conference* (Issue 2).
- Bergstra, J., Ca, J. B., & Ca, Y. B. (2012). Random Search for Hyper-Parameter Optimization Yoshua Bengio. In *Journal of Machine Learning Research* (Vol. 13). <http://scikit-learn.sourceforge.net>

Bharaj, G., et al. (2015). Computational design of metallophone contact sounds. *ACM Transactions on Graphics (TOG)*, 34(6), 1–13. <https://doi.org/10.1145/2816795.2818137>

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140. <https://doi.org/10.1007/BF00058655>

Candelieri, A., Perego, R., & Archetti, F. (2018). Bayesian optimization of pump operations in water distribution systems. *Journal of Global Optimization*, 71(1), 213–235. <https://doi.org/10.1007/s10898-018-0641-2>

Chabanet, S., Bril El-Haouzi, H., & Thomas, P. (2021). Coupling digital simulation and machine learning metamodel through an active learning approach in Industry 4.0 context. *Computers in Industry*, 133. <https://doi.org/10.1016/j.compind.2021.103529>

De La Fuente, R., Erazo, I., & Smith, R. L. (2019). Enabling intelligent processes in simulation utilizing the tensorflow deep learning resources. *Proceedings - Winter Simulation Conference*, 2018-Decem, 1108–1119. <https://doi.org/10.1109/WSC.2018.8632539>

dos Santos, C. H., Gabriel, G. T., do Amaral, J. V. S., Montevechi, J. A. B., & de Queiroz, J. A. (2021). Decision-making in a fast fashion company in the Industry 4.0 era: a Digital Twin proposal to support operational planning. *The International Journal of Advanced Manufacturing Technology*, 116(5–6), 1653–1666. <https://doi.org/10.1007/s00170-021-07543-z>

Fan, Q., & Hu, J. (2018). Surrogate-based promising area search for Lipschitz continuous simulation optimization. *INFORMS Journal on Computing*, 30(4), 677–693. <https://doi.org/10.1287/ijoc.2017.0801>

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4), 367–378. [https://doi.org/10.1016/S0167-9473\(01\)00065-2](https://doi.org/10.1016/S0167-9473(01)00065-2)

Ganjisaffar, Y., Caruana, R., & Lopes, C. V. (2011). Bagging gradient-boosted trees for high precision, low variance ranking models. *SIGIR'11 - Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, c, 85–94. <https://doi.org/10.1145/2009916.2009932>

Guimarães, A. M. C., Leal, J. E., & Mendes, P. (2018). Discrete-event simulation software selection for manufacturing based on the maturity model. *Computers in Industry*, 103, 14–27. <https://doi.org/10.1016/j.compind.2018.09.005>

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. In Springer (Vol. 2). Springer New York. <https://doi.org/10.1007/978-0-387-84858-7>

Law, A. M. (2013). *Simulation Modeling and Analysis* (5th ed.). McGraw-Hill. <https://www.jstor.org/stable/2288169?origin=crossref>

Louk, M. H. L., & Tama, B. A. (2023). Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Systems with Applications*, 213. <https://doi.org/10.1016/j.eswa.2022.119030>

Moghaddam, S., & Mahlooji, H. (2017). A new metamodel-based method for solving semi-expensive simulation optimization problems. *Communications in Statistics: Simulation and Computation*, 46(6), 4795–4811. <https://doi.org/10.1080/03610918.2015.1134567>

Montevecchi, J. A. B., Amaral, J. V. S., Miranda, R. C., Dos Santos, C. H., Brito, F. O. D., & Machado, M. E. F. H. S. (2024). Ensemble-Based Infill Search Simulation Optimization *Framework*. In *Proceedings of the Winter Simulation Conference*, pp. 3059–3070. <https://doi.org/10.1109/wsc60868.2023.10408380>

Parnianifard, A., Azfanizam, A. S., Ariffin, M. K. A., Ismail, M. I. S., & Ale Ebrahim, N. (2019). Recent developments in metamodel based robust black-box simulation optimization: An overview. *Decision Science Letters*, 8(1), 17–44. <https://doi.org/10.5267/j.dsl.2018.5.004>

Rodič, B. (2017). Industry 4.0 and the New Simulation Modelling Paradigm. *Organizacija*, 50(3), 193–207. <https://doi.org/10.1515/orga-2017-0017>

Rojas-Gonzalez, S., & Van Nieuwenhuyse, I. (2020). A survey on kriging-based infill algorithms for multiobjective simulation optimization. *Computers and Operations Research*, 116, 104869. <https://doi.org/10.1016/j.cor.2019.104869>

Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *Stata Journal*, 20(1), 3–29. <https://doi.org/10.1177/1536867X20909688>

Sousa Junior, W. T. de, Montevecchi, J. A. B., Miranda, R. de C., & Campos, A. T. (2019). Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. *Computers & Industrial Engineering*, 128(December 2018), 526–540. <https://doi.org/10.1016/j.cie.2018.12.073>

Sousa Junior, W. T. de, Montevecchi, J. A. B., Miranda, R. de C., Oliveira, M. L. M. de, & Campos, A. T. (2020). Shop floor simulation optimization using machine learning to improve parallel metaheuristics. *Expert Systems with Applications*, 150, 113272. <https://doi.org/10.1016/j.eswa.2020.113272>

Xu, J., Huang, E., Hsieh, L., Lee, L. H., Jia, Q. S., & Chen, C. H. (2016). Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. *Journal of Simulation*, 10(4), 310–320. <https://doi.org/10.1057/s41273-016-0037-6>

Yin, R., Liu, Z., & Zheng, N. (2022). A Simulation-Based Model for Continuous Network Design Problem Using Bayesian Optimization. *IEEE Transactions on Intelligent Transportation Systems*, 23(11), 20352–20367. <https://doi.org/10.1109/T>

Yousefi, M., & Yousefi, M. (2019). Human resource allocation in an emergency department: A metamodel-based simulation optimization. *Kybernetes*, 49(3), 779–796. <https://doi.org/10.1108/K-02-2018-0074>

Zhang, Y., & Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 58, 308–324. <https://doi.org/10.1016/j.trc.2015.02.019>