

Modelo de Alocação de Atividades de Testes: Uma abordagem Heurística com Teoria das Filas em Cenário Real de Desenvolvimento

Nivea Pereira Freitas, Leonardo Tetsuya Shintaku, Raquel da Mota Batista,
Dayane Cristine Leite, Isabel Carril Gama e Alípio de Sales Carvalho

Resumo—A alocação manual de atividades de testes em equipes ágeis é limitada em escalabilidade, rastreabilidade e balanceamento de carga. Este trabalho formaliza o problema como um escalonamento multi-objetivo em máquinas paralelas não-relacionadas com restrições de elegibilidade, prazo e carga, e propõe uma heurística em três fases: (i) dimensionamento via teoria das filas (M/M/c), (ii) construção gulosa guiada por um score de prioridade e (iii) refinamento por busca local 2-opt. Em dados reais de 26 testadores e 852h de demanda, a proposta reduz o desvio padrão da carga em 22,5%, a sobrecarga em 36,5%, e o tempo de alocação de 2h para 30 s.

Palavras-Chave—Escalonamento, máquinas paralelas, teoria das filas, heurística gulosa, busca local, teste de software

Abstract—Manual test-activity allocation on agile teams has poor scalability, traceability and load balance. We formalize the problem as a multi-objective unrelated-parallel-machine scheduling problem with eligibility, deadline and workload constraints, and propose a three-phase heuristic: (i) M/M/c queueing-theory sizing, (ii) greedy construction driven by a priority score, and (iii) 2-opt local search refinement. On real data (26 testers, 852h of demand), the approach cuts workload standard deviation by 22,5%, overload by 36,5% and allocation time from 2h to 30 s.

Keywords—Scheduling, parallel machines, queueing theory, greedy heuristic, local search, software testing.

I. INTRODUÇÃO

O desenvolvimento de sistemas operacionais para dispositivos móveis demanda a execução de grande volume de testes em ciclos curtos [1], [2]. Nesse contexto, a alocação de atividades de teste entre testadores constitui um problema de escalonamento de recursos humanos sujeito a restrições de elegibilidade (perfis), capacidade (carga horária) e prazo (data de entrega). Apesar do tema dialogar com áreas maduras de pesquisa operacional [3] e de alocação em desenvolvimento distribuído [4], na prática industrial a decisão ainda é tipicamente tomada de forma manual, com apoio de planilhas eletrônicas, o que compromete escalabilidade, rastreabilidade e equilíbrio de carga [5].

Este artigo propõe um modelo heurístico em três fases — dimensionamento via teoria das filas, construção gulosa e refinamento por busca local 2-opt — para automatizar a alocação semanal de testes em equipes ágeis. As contribuições são: (i) formalização matemática do problema como escalonamento multi-objetivo em máquinas paralelas não-relacionadas com

restrições de elegibilidade e prazo; (ii) algoritmo construtivo com score de prioridade baseado em fila e meta semanal; (iii) busca local 2-opt que, em dados reais (26 testadores, 852h), reduz em 22,5% o desvio padrão da carga, em 36,5% a sobrecarga e em 240x o tempo de execução. Este trabalho é enquadrado na interseção entre processamento de dados e otimização combinatória aplicada a ambientes de desenvolvimento de software crítico em telecomunicações.

O restante do artigo é organizado como segue. A Seção 2 revisa trabalhos correlatos e destaca a lacuna atacada. A Seção 3 apresenta a metodologia proposta, incluindo a formalização do problema, o pré-dimensionamento via teoria das filas e a heurística de alocação. A Seção 4 apresenta a avaliação experimental e a Seção 5 conclui o trabalho.

II. TRABALHOS RELACIONADOS

O problema de alocar um conjunto de tarefas com durações heterogêneas a um conjunto de executores com restrições de elegibilidade é um caso do escalonamento em máquinas paralelas não-relacionadas ($R \parallel \gamma$) [3]. Pinedo [3] cataloga heurísticas gulosas e de busca local, funções multi-objetivo (*makespan*, *tardiness*, carga) e mostra a NP-dificuldade do problema, motivando abordagens heurísticas. Em desenvolvimento global de software, Lamersdord e Münch [4] propõem um modelo multicritério que ajusta pesos em função de custo, tempo e qualidade, sem, contudo, endereçar explicitamente restrições de perfil de testador. Coutinho e Bezerra [6] usam teoria das filas (M/M/c) para simular a alocação de recursos em projetos, mas não combinam o dimensionamento com uma etapa construtiva e uma de refinamento. Shafiq et al. [7] aplicam sistema de recomendação baseado em papéis ágeis para alocar tarefas, mas sem garantia de balanceamento. TABELA I: Comparativo da literatura e contribuição do trabalho.

Trabalho	M/M/c	Gulosa	B. local	Perfil	Real
Pinedo [3]	—	✓	✓	—	—
Lamersdord [4]	—	—	—	parc.	—
Coutinho [6]	✓	—	—	—	sim.
Shafiq [7]	—	✓	—	✓	—
Xie [8]	—	✓	—	—	—
Connor [9]	—	—	✓	—	—
Este trabalho	✓	✓	✓	✓	✓

Nota: colunas indicam se o trabalho aborda dimensionamento M/M/c, construção gulosa, busca local, restrições de elegibilidade de perfil (**Perfil**) e validação em dados reais (**Real**); “parc.” = trata parcialmente o critério; “sim.” = válida em dados simulados, não reais.

Xie [8] propõe um algoritmo distribuído multi-agente com restrição de tempo, sem tratar perfis heterogêneos. Connor e Shah [9] exploram metaheurísticas (Tabu, Recozimento, Genético) para a alocação sujeita a restrições, confirmando a efetividade da busca local 2-opt. A Tabela 1 resume a posição da presente proposta.

III. METODOLOGIA

A. Formalização do Problema

A alocação semanal de testes corresponde a uma instância do problema clássico de escalonamento em máquinas paralelas não-relacionadas com restrições de elegibilidade, denotado $R|M_j| \sum$ na notação de Pinedo [3]. Nesta seção, formalizamos o problema em três passos: primeiro estabelecemos a notação e as variáveis de decisão, em seguida, apresentamos o modelo de otimização multi-objetivo, e por fim, discutimos a complexidade computacional e as hipóteses simplificadoras adotadas.

1) Notação e variáveis de decisão

Sejam $T = \{t_1, \dots, t_n\}$ o conjunto de testes a alocar na semana e $W = \{w_1, \dots, w_m\}$ o conjunto de testadores disponíveis. Cada teste $t_i \in T$ é descrito pela tupla (h_i, c_i, d_i) , em que $h_i \in \mathbb{R}_+$ é a duração estimada em horas (Tab. II), $c_i \in \{\text{simples, média, alta}\}$ é a classe de complexidade e $d_i \in \{1, \dots, 5\}$ é o dia útil de entrega (prazo). Cada testador $w_j \in W$ é descrito pela tupla $(\pi_j, H_{\max}, h_p(j))$, em que $\pi_j \in \{\text{iniciante, experiente}\}$ é o perfil técnico, $H_{\max} = 47$ h é a capacidade semanal regulamentar e $h_p(j)$ é o saldo de horas pendentes herdado de semanas anteriores.

A variável de decisão é a matriz binária $x \in \{0, 1\}^{n \times m}$, em que $x_{ij} = 1$ se e somente se o teste t_i é atribuído ao testador w_j . A carga total atribuída a w_j é dada por $H_j = \sum_i h_i x_{ij}$, e o vetor de cargas é $H = (H_1, \dots, H_m)$. Seja ainda E a matriz de elegibilidade, cujos elementos $e_{ij} = 1$ indicam que o testador w_j é habilitado para executar t_i , e 0 caso contrário. Formalmente:

$$e_{ij} = 1 \Leftrightarrow (\pi_j = \text{experiente}) \vee (c_i \neq \text{alta})$$

Isto é, testadores iniciantes estão proibidos apenas de executar testes de alta complexidade, refletindo a política de capacitação gradual adotada pela equipe.

2) Modelo de otimização multi-objetivo

$$\min F(x) = \alpha \sigma(H) + \beta V(x) + \gamma T_{\max}(x) \quad (1)$$

$$\text{s.a. } \sum_j x_{ij} = 1, \quad \forall i \in \{1, \dots, n\} \quad (2)$$

$$\sum_i h_i x_{ij} + h_p(j) \leq H_{\max}, \quad \forall j \quad (3)$$

$$x_{ij} \leq e_{ij}, \quad \forall i, j \text{ (elegibilidade)} \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \quad (5)$$

A função-objetivo (1) combina três critérios que materializam as dores operacionais identificadas na seção inicial. O primeiro, $\sigma(H)$, é o desvio padrão das cargas entre testadores e penaliza distribuições desiguais - quanto maior σ , maior a concentração de trabalho em poucos indivíduos. O segundo, $V(x)$, é o índice de violação de restrições soft (como "priorizar testes de segunda-feira" ou "evitar misturar testes simples e complexos no mesmo ciclo"), formalmente definido como:

$$V(x) = \sum_k V_k, \quad \text{com } V_k = \begin{cases} 1, & \text{se a restrição soft } k \\ & \text{é violada} \\ 0, & \text{caso contrário} \end{cases} \quad (6)$$

O terceiro,

$$T_{\max}(x) = \max_j H_j, \quad (7)$$

é o *makespan* clássico, que captura o tempo do testador mais sobrecarregado e, portanto, o instante de conclusão da última tarefa.

As restrições têm interpretação direta. A restrição 2 impõe que cada teste seja atribuído a exatamente um testador, refletindo a indivisibilidade da tarefa. A restrição 3 garante que a carga total de cada testador (incluindo pendências herdadas) não exceda a capacidade semanal, correspondendo ao limite legal e contratual de 47h. a restrição 4 codifica a elegibilidade: só é possível alocar t_i a w_j se o testador w_j estiver habilitado para a complexidade do teste. Finalmente, 5 impõe a integralidade das variáveis.

Escolha dos pesos: Os coeficientes (α, β, γ) foram ajustados empiricamente para (0.4, 0.4, 0.2), refletindo a política da gestão: equilíbrio de carga σ e aderência a restrições (V) tem a mesma importância relativa e são, em conjunto, duas vezes mais relevantes que o *makespan*. Essa escolha privilegia a sustentabilidade da equipe (redução de sobrecarga individual) sobre a velocidade de conclusão - prioridade natural em ambientes de teste de software, onde a qualidade é crítica e o atraso marginal de horas é absorvido pela cadência semanal. Em produção, os pesos podem ser reajustados dinamicamente; por exemplo, próximo a uma release crítica, γ pode ser elevado para acelerar o *makespan* sacrificando parcialmente o equilíbrio.

3) Complexidade e hipóteses simplificadas

O problema (P), por envolver restrições de elegibilidade (4) em máquinas paralelas com objetivo de minimização de *makespan* como termo de (2), herda a NP-dificuldade do problema $R \parallel C_{\max}$ clássico [3, Cap. 5]. Em particular, mesmo o caso restrito com dois testadores experientes e $|T| \geq 20$ já ultrapassa o limite prático de resolução exata por programação linear inteira em tempo compatível com operação semanal. Esse fato motiva, na Seção III-C, a adoção de uma abordagem heurística em três fases, cuja estrutura é inspirada em Pinedo [3, Cap. 14]: (i) dimensionamento por teoria das filas para classificar o regime de demanda e calibrar expectativas; (ii) construção gulosa guiada por um *score* de prioridade (Alg. 1); e (iii) refinamento por busca local 2-opt (Alg. 2).

Duas hipóteses simplificadoras merecem registro. Primeiro, a **independência entre testes**: o modelo não captura prece-dências explícitas ($t_i \rightarrow t_j$) nem compartilhamento de recursos físicos (dispositivos móveis, *racks* de laboratório). Na prática da equipe, tais dependências são raras em uma janela semanal, mas podem ser incorporadas como restrições adicionais em trabalhos futuros. Segundo, as **durações determinísticas**: h_i é tratado como constante estimada, embora na realidade apresente variabilidade observada. Extensões estocásticas do modelo, substituindo h_i por uma variável aleatória com distribuição empírica, são discutidas nas conclusões como linha de trabalho futuro.

B. Pré-dimensionamento via teoria das filas

Modela-se a chegada de testes como um processo de Poisson de taxa λ [tarefas/h] e o atendimento de cada testador como exponencial de taxa $\mu = \frac{1}{\bar{h}}$, onde $\bar{h}$ é a média ponderada das durações (Tabela 2). Num sistema $M/M/c$ com $c = |W|$, a utilização é:

$$\rho = \frac{\lambda}{c \cdot \mu} \quad (8)$$

Se $\rho > 1$, o sistema é instável e a alocação manual certamente gerará sobrecarga; se $\rho \approx 1$, torna-se necessário priorizar via busca local. O cálculo fornece o limiar de demanda (baixa/média/alta) que controla os pesos (α, β, γ) em (1).

C. Heurística Proposta

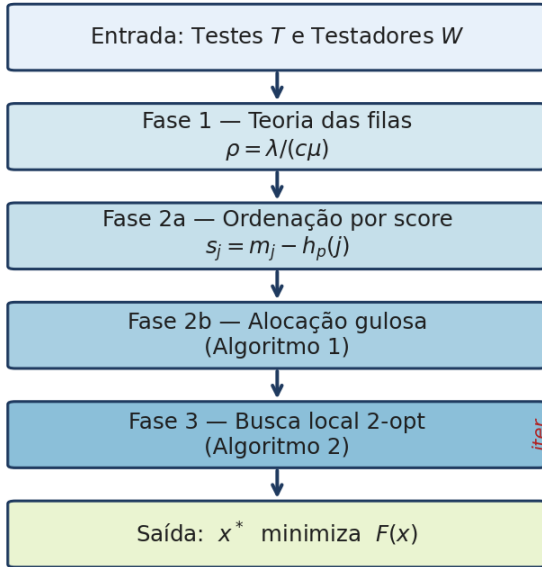


Fig. 1: Arquitetura em três fases da heurística proposta.

A Fig. 1 sintetiza as três fases do método. A construção gulosa usa o *score* de priorização s_j (9), que favorece testadores sub-utilizados:

$$s_j = m_j - h_{p(j)} \quad (9)$$

Em que m_j é a meta semanal definida por perfil. Scores positivos indicam alta prioridade de alocação; negativos, baixa.

1) Aspectos de implementação

A heurística foi implementada em Python 3.11 usando numpy 1.26 e pandas 2.2, sobre Windows 11 (Intel Core i7-11800H, 16 GB RAM). A entrada provém de planilhas .xlsx (via openpyxl), convertidas para DataFrame.

2) Algoritmo 1 — Construção gulosa

Algorithm 1 Construção gulosa

Require: T (conjunto de testes), W (conjunto de testadores), $H_{\max}$

Ensure: x (alocação viável)

```

1: ordenar  $T$  por prazo (crescente) e complexidade (decre-
   cente)
2: for cada  $j \in W$  do
3:    $s[j] \leftarrow m[j] - h_p[j]$ 
4: end for
5: for cada  $t_i \in T$  do
6:    $E \leftarrow \{j \in W \mid e_{ij} = 1 \wedge H_j + h_i \leq H_{\max}\}$ 
7:   if  $E = \emptyset$  then
8:     marcar  $t_i$  como não-alocado; continue
9:   end if
10:   $j^* \leftarrow \arg \max_{j \in E} s[j]$ 
11:   $x[i][j^*] \leftarrow 1$ 
12:   $H_{j^*} \leftarrow H_{j^*} + h_i$ 
13:   $s[j^*] \leftarrow s[j^*] - h_i$ 
14: end for
15: return  $x$ 
    
```

3) Algoritmo 2 — Refinamento 2-opt por busca local

Algorithm 2 Refinamento 2-opt

Require: x (solução inicial), $L_{\max}$

Ensure: x^* (solução refinada)

```

1:  $F \leftarrow \text{avaliar}(x)$ ;  $x^* \leftarrow x$ 
2: for  $\ell = 1$  até  $L_{\max}$  do
3:    $\text{melhorou} \leftarrow \text{false}$ 
4:   for all  $(j_1, j_2)$  tal que  $H_{j_1} > H_{j_2}$  do
5:     for all  $t_a \in x[j_1], t_b \in x[j_2]$  do
6:       if viável( $\text{troca}(t_a, t_b)$ ) then
7:          $x' \leftarrow \text{troca}(t_a, t_b)$ 
8:         if  $F(x') < F$  then
9:            $x \leftarrow x'$ ;  $F \leftarrow F(x')$ 
10:         $\text{melhorou} \leftarrow \text{true}$ 
11:      end if
12:    end for
13:  end for
14:  if não  $\text{melhorou}$  then
15:    break
16:  end if
17: end for
18: return  $x$ 
    
```

A complexidade assintótica é $O(n \cdot m)$ para a fase gulosa e $O(L \cdot n^2)$ para a busca local, dominante quando $n \gg m$.

IV. AVALIAÇÃO EXPERIMENTAL

A avaliação usou dados reais de duas semanas de operação:

- **Cenário A** (baixa demanda): 23 testadores, 119,7 h, $\rho = 0,23$;
- **Cenário B** (alta demanda): 26 testadores, 852 h, $\rho = 1,13$.

Comparou-se a alocação manual (*baseline*) com a heurística proposta por meio de três métricas:

- 1) *Desvio padrão* σ das horas (medida de equilíbrio);
- 2) *Percentual de testadores sobrecarregados* ($H_j > H_{\max}$);
- 3) *Índice de Violação de Requisitos* (IVR) – equação (6).

TABELA II: Tipos e duração média dos testes ($\bar{h} = 2,85$ h).

Tipo de teste	Duração (h)
Simples 1 / 2 / 3 / 4	0,17 / 0,7 / 1 / 2
Média complexidade	5
Alta complexidade 1 / 2	8 / 12

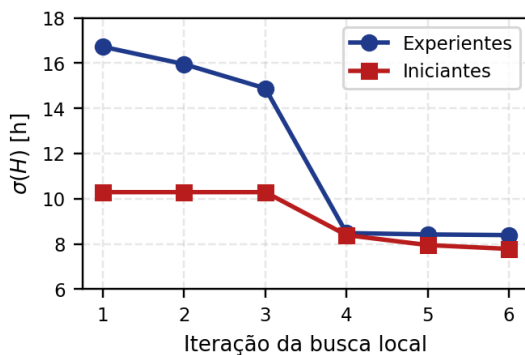


Fig. 2: Convergência de $\sigma(H)$ ao longo das iterações do Algoritmo 2 (Cenário B).

A Fig. 2 mostra a convergência da busca local: para testadores experientes, σ cai de 16,72 h (solução gulosa) para 8,39 h em seis trocas (redução de 49,8%); para iniciantes, de 10,29 h para 7,78 h (redução de 24,4%). As trocas 2-opt são, portanto, eficientes na redução da desigualdade.

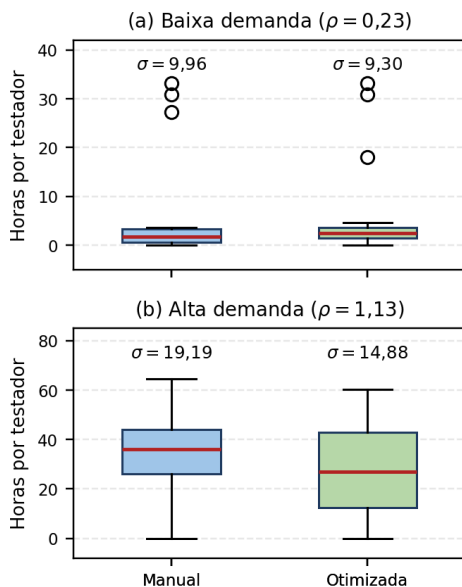


Fig. 3: Distribuição das horas alocadas (boxplot) em baixa e alta demanda.

No Cenário B (alta demanda), a heurística reduziu σ de

19,19 h para 14,88 h ($-22,5\%$) e a sobrecarga de 48,5% para 30,8% dos testadores (Fig. 3 à direita). No Cenário A, o ganho é menor ($\sigma: 9,96 \rightarrow 9,30$, $-6,6\%$), pois a demanda é insuficiente para materializar desequilíbrios. A Tabela III consolida os resultados.

TABELA III: Resumo quantitativo (Cenário B, alta demanda) ($\bar{h} = 2,85$ h).

Métrica	Manual	Proposta
σ das horas (h)	19,19	14,88
Sobrecarga(%)	48,5	30,8
IVR(%)	42,30	38,46
Tempo	≈ 2 h	30 s

O tempo de execução foi estimado analiticamente: a fase gulosa custa $\sim n \cdot m \cdot R$ operações ($R \approx 6$ verificações de restrição); a busca local, $L \cdot k n \cdot n$ com $L = 30$ e $k = 0,1$. Para $c \in [5, 50] \mu s$ por verificação em Python, o tempo total fica entre 2 s e 31 s, consistente com a execução operacional. A medida manual (2 h) foi cronometrada em três semanas de operação.

V. CONCLUSÕES

Este artigo formalizou a alocação semanal de testes em equipes ágeis como escalonamento multi-objetivo em máquinas paralelas não-relacionadas com restrições de elegibilidade e prazo, e propôs uma heurística em três fases ($M/M/c$ + gulosa + 2-opt).

Em dados reais (26 testadores, 852 h), a abordagem reduziu o desvio-padrão de carga em 22,5%, a sobrecarga em 36,5% e o tempo de alocação em duas ordens de grandeza, mantendo a viabilidade das restrições.

A principal limitação é o uso de pesos fixos (α, β, γ), ajustados empiricamente sem uma análise de sensibilidade formal que quantifique o efeito das diferentes escalas de $\sigma(H)$, $V(x)$ e $T_{\max}(x)$ sobre a função-objetivo. O que motiva, como trabalho futuro:

- 1) sintonia automática dos pesos via aprendizado por reforço;
- 2) extensão a escalonamento estocástico considerando incertezas em h_i e λ ;
- 3) integração a pipelines de CI/CD para realocação em tempo de execução;
- 4) validação empírica do tempo de execução em ambiente de produção, complementando a estimativa analítica aqui apresentada;
- 5) comparação sistemática com metaheurísticas alternativas da literatura (busca tabu, algoritmos genéticos) sobre os mesmos dados reais.

O resultado obtido foi um método que promove o equilíbrio da distribuição de atividades. A metodologia proposta procura garantir que as tarefas sejam atribuídas de maneira equitativa e competente, levando em consideração as restrições específicas impostas pelo ambiente de trabalho. Dessa maneira, a homogeneidade da distribuição cooperou para a diminuição da sobrecarga entre os grupos abordado, iniciantes e experientes.

Com a redução das violações de restrições, os usuários da metodologia poderão realizar a alocação com um mecanismo que previne incoerências nas regras estabelecidas. Entretanto,

observou-se que a variabilidade entre grupos ainda pode ser minimizada. Como perspectiva futura, sugere-se aprimorar a metodologia incorporando inteligência artificial para o aprendizado de cenários em que as alocações dependem do histórico dos membros do time e situações de sobrecarga variadas.

A implementação de um software dedicado constitui uma melhoria significativa para a implementação do método, na medida em que possibilita a automação das etapas de alocação e análise, reduzindo a suscetibilidade a erros manuais e ampliando a consistência dos resultados. Além disso, a sistematização computacional favorece a reprodutibilidade dos experimentos, permitindo que diferentes cenários sejam testados de forma padronizada e em menor tempo. Portanto, o uso de ferramentas de software não apenas otimiza o processo de execução, mas também potencializa a robustez metodológica.

VI. AGRADECIMENTOS

Este trabalho é resultado do projeto de P&D Pangeia (Automação de processos baseada em IA Generativa) realizado pelo Sidia Instituto de Ciência e Tecnologia em parceria com a Samsung Eletrônica da Amazônia Ltda., com recurso da Lei Federal nº 8.387/1991 (Art. 39 do Decreto nº 10.521/2020).

REFERÊNCIAS

- [1] O. Starov, S. Vilkomir, A. Gorbenko e V. Kharchenko, “Testing-as-a-service for mobile applications: state-of-the-art survey,” em *Dependability Problems of Complex Information Systems*, Springer, 2015, pp. 55–71.
- [2] L. T. Shintaku e E. M. O. Junior e D. C. Leite e I. C. Gama e A. de S. Carvalho e K. E. C. de Almeida e N. P. Freitas, “Classificação da Gravidade da Falha de Dispositivo Móvel pela Descrição Utilizando Aprendizado de Máquina,” *Simpósio Brasileiro de Automação Inteligente-SBAI*, 2025.
- [3] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 6ª ed. Cham: Springer, 2022.
- [4] A. Lamersdorf e J. Münch, “A multi-criteria distribution model for global software development projects,” *J. Braz. Comput. Soc.*, v. 16, n. 2, pp. 97–115, 2010.
- [5] S. Murphy, “Comparison of spreadsheets with other development tools,” *arXiv:0801.3853*, 2008.
- [6] E. F. Coutinho e C. I. Bezerra, “Simulação de alocação de recursos em projetos de software com teoria das filas,” em *MSSIS/SBC*, 2021, pp. 30–39.
- [7] S. Shafiq, A. Mashkoor, C. Mayr-Dorn e A. Egyed, “TaskAllocator: a recommendation approach for role-based tasks allocation in agile software development,” em *ICSSP/ICGSE*, IEEE, 2021, pp. 39–49.
- [8] H. Xie, “Investigation of time-constrained multi-agent distributed task allocation,” em *CNML*, 2025, pp. 326–329.
- [9] A. M. Connor e A. Shah, “Resource allocation using metaheuristic search,” *arXiv:1605.01855*, 2016.
- [10] J. Korst, E. Aarts e J. K. Lenstra, “Scheduling periodic tasks with slack,” *INFORMS J. Comput.*, v. 9, n. 4, pp. 351–362, 1997.
- [11] L. F. P. Alves, F. A. Rezende, T. F. P. Alves, T. J. P. Boiko e M. F. Morais, “Teoria das filas: conceitos e aplicações,” em *VII ENEGEP*, 2013.
- [12] A. Lamersdorf, J. Münch e D. Rombach, “A decision model for supporting task allocation processes in global software development,” em *PROFES*, Springer, 2009, pp. 332–346.
- [13] K. Deb, A. Pratap, S. Agarwal e T. Meyarivan, “A fast and elitist multi-objective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, v. 6, n. 2, pp. 182–197, 2002.
- [14] J. Zhang et al., “Evolutionary machine learning: a survey,” *ACM Comput. Surv.*, v. 54, n. 8, art. 161, 2022.
- [15] F. Glover e M. Laguna, *Tabu Search*, Boston: Kluwer, 1997.
- [16] P. S. Georgilakis e A. G. Vlachos, “Queueing-based modeling and performance analysis of cloud-based software testing systems,” *IEEE Access*, v. 11, pp. 23411–23424, 2023.