57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

FORMULAÇÕES PARA O PROBLEMA DE SEQUENCIAMENTO DE TAREFAS COM SETUPS DEPENDENTES DA SEQUÊNCIA E RESTRIÇÕES DE INVENTÁRIO

Rafael Morais, Teobaldo Bulhões, Anand Subramanian
Universidade Federal da Paraíba
Centro de Informática, Universidade Federal da Paraíba (UFPB) – R. dos Escoteiros, s/n – Mangabeira, João Pessoa – PB, CEP 58058-600
rafaelsobralm@gmail.com, tbulhoes@ci.ufpb.br, anand@ci.ufpb.br

RESUMO

Este trabalho aborda o problema de minimização do *makespan* no sequenciamento de tarefas em máquina única, considerando datas de liberação, tempos de *setup* dependentes da sequência e restrições de inventário. Para representar o problema, foram propostas quatro formulações matemáticas distintas: a primeira baseada em variáveis indexadas por posição; a segunda, por arcos; a terceira, por arcos combinados com níveis de inventário; e a quarta, por arcos indexados no tempo. A fim de avaliar o desempenho de cada modelo, conduziram-se experimentos sobre um conjunto de 450 instâncias geradas. Os resultados indicam que a formulação baseada em arcos apresentou o melhor desempenho geral, sendo a única capaz de resolver instâncias com até 50 tarefas e alcançando a solução ótima em 154 casos.

PALAVRAS CHAVE. Sequenciamento. Inventário. Programação linear inteira.

Tópicos (Otimização Combinatória, Programação Matemática)

ABSTRACT

This work addresses the problem of minimizing makespan in single-machine job scheduling, considering release dates, sequence-dependent setup times, and inventory constraints. Four distinct mathematical formulations were proposed to represent the problem: the first based on position-indexed variables; the second on arc-indexed variables; the third on arcs combined with inventory levels; and the fourth on arc-time indexes. To evaluate the performance of each model, experiments were conducted on a set of 450 generated instances. The results indicate that the arc-indexed formulation showed the best overall performance, being the only one capable of solving instances with up to 50 tasks and achieving the optimal solution in 154 cases.

KEYWORDS. Scheduling. Inventory. Integer linear programming.

Paper topics (Combinatorial Optimization, Mathematical Programming)

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

1. Introdução

Problemas de sequenciamento de tarefas desempenham um papel fundamental na gestão de operações e manufatura, especialmente em ambientes onde a eficiência e a coordenação de recursos são essenciais. O impacto do *scheduling* na produtividade em setores como indústria e serviços faz com que esse tipo de problema seja amplamente estudado, e dentre suas diferentes configurações, a variante que considera máquina única é uma das mais relevantes, uma vez que problemas em ambientes mais complexos frequentemente podem ser decompostos em problemas dessa classe [Pinedo, 2022].

No entanto, muitas aplicações do mundo real envolvem características adicionais que podem incrementar significativamente a complexidade do problema de sequenciamento. Duas dessas características são os tempos de preparação dependentes da sequência e as restrições de inventário.

Tempos de preparação dependentes da sequência são comuns em ambientes industriais nos quais a troca entre diferentes tarefas requer processos como limpeza, reconfiguração ou ajuste de máquinas, e o tempo ou custo necessário para essa troca varia conforme o par de tarefas envolvido. Os tempos de *setup* muitas vezes são desprezados para facilitar a resolução do problema, porém sua inclusão pode levar a grande economia em muitos casos [Allahverdi et al., 2008].

Por outro lado, as restrições de inventário são fundamentais em sistemas onde produtos intermediários precisam ser armazenados temporariamente antes de serem entregues ou processados, e há limites para a capacidade ou disponibilidade de armazenamento. Assim, a presença simultânea de tempos de *setup* dependentes da sequência e de restrições de inventário impõe o desafio de encontrar soluções que conciliem eficiência nos preparos com a viabilidade do controle do inventário.

Este trabalho apresenta problema de minimização do *makespan* no sequenciamento de tarefas em máquina única, considerando datas de liberação, tempos de *setup* dependentes da sequência e restrições de inventário. Na notação clássica de Graham et al. [1979], o problema é denotado por $1|r_j, s_{ij}, inv|C_{max}$. Como principal contribuição, são propostas diferentes formulações matemáticas para o problema, as quais são avaliadas por meio de experimentos computacionais realizados sobre um conjunto de instâncias geradas.

A Seção 2 apresenta uma breve revisão da literatura relacionada. O problema abordado é formalmente definido na Seção 3. As formulações propostas são detalhadas na Seção 4. Os experimentos computacionais, bem como os resultados obtidos, são discutidos na Seção 5. Por fim, a Seção 6 reúne as considerações finais deste estudo.

2. Revisão Bibliográfica

Embora a literatura sobre problemas de scheduling seja ampla, há uma quantidade limitada de estudos que abordam variantes com restrições de inventário. Como destacado por Davari et al. [2020], essas restrições podem ser vistas como uma generalização dos recursos não renováveis — recursos que são consumidos durante a execução das tarefas e não são reabastecidos após sua utilização. Exemplos típicos incluem matéria-prima, combustível, energia ou recursos financeiros, conforme discutido em trabalhos como Carlier e Kan [1982], Slowiński [1984], Toker et al. [1991].

Os estudos de Briskorn et al. [2010]; Briskorn e Leung [2013]; Briskorn e Pesch [2013] abordaram o problema de *scheduling* em máquina única com restrições de inventário sob diferentes perspectivas. Inicialmente, Briskorn et al. [2010] analisaram a complexidade de variantes do problema com inventário ilimitado para diferentes funções objetivo. Posteriormente, Briskorn e Leung [2013] propuseram um algoritmo *branch-and-bound* (B&B) para minimizar o atraso ponderado, conseguindo resolver instâncias com até 240 tarefas ao aplicar estratégias eficientes para cálculo de *bounds*. Já Briskorn e Pesch [2013] exploraram o problema no contexto de um terminal de transbordo, aplicando abordagens de *variable neighborhood search* (VNS) e *variable neighborhood descent* (VND) para funções objetivo diversas, destacando o VND como mais preciso, embora mais custoso computacionalmente.

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

O trabalho de Ghorbanzadeh et al. [2019] investigou o problema de sequenciamento de operações de carga e descarga em um terminal com uma única estação, modelando-o como um problema de scheduling em máquina única com datas de liberação e restrições de inventário, formalmente representado por $1|r_j, s_{ij}|C_{\max}$. Como métodos de solução, os autores propuseram um algoritmo B&B e um algoritmo de programação dinâmica, ambos capazes de resolver instâncias com até 30 tarefas.

De forma semelhante, Davari et al. [2020] estudaram uma variante do mesmo problema, para a qual apresentaram duas formulações de programação inteira: uma com variáveis de decisão indexadas por arco e tempo, e outra baseada na posição das tarefas na sequência. Adicionalmente, foi proposto um algoritmo exato do tipo *guess-and-check* (GC), avaliado em um conjunto de 900 instâncias, com tamanhos variando entre 10 e 100 tarefas, geradas pelos próprios autores.

Esse mesmo conjunto de instâncias foi posteriormente utilizado por Guerra et al. [2024], que propuseram uma série de algoritmos populacionais para o problema, obtendo melhorias em relação a alguns dos resultados apresentados por Davari et al. [2020].

O problema de minimizar o makespan com datas de liberação e tempos de *setup* dependentes da sequência foi explorado por Montoya-Torres et al. [2012], que propuseram uma heurística baseada em inserções aleatórias tanto para o caso de uma única máquina quanto para máquinas paralelas. Nessa abordagem, assume-se que o tempo de *setup* da primeira tarefa é inexistente, ou seja, a máquina está inicialmente preparada para processar qualquer tarefa.

Complementando essa linha de pesquisa, Velez-Gallego et al. [2016] apresentaram uma modelagem por programação inteira mista (MILP) e um algoritmo *beam search* (BS) para o mesmo problema em ambiente de máquina única, considerando *release dates* e *setups* dependentes da sequência, com a premissa de setup não antecipatório, e incluindo o *setup* inicial. Mais recentemente, Fan et al. [2019] propuseram uma nova abordagem heurística, denominada *variable block insertion heuristic* (VBIH), também voltada para esse problema. Por fim, Morais et al. [2024] desenvolveram um algoritmo *branch-and-price* (BP) e uma heurística híbrida que combina *iterated local search* (ILS) com BS para o problema $1|r_j, s_{ij}|C_{max}$.

3. Definição do problema

Seja $J = \{1, 2, \ldots, n\}$ o conjunto de n tarefas a serem processadas em uma única máquina. Cada tarefa $j \in J$ possui um tempo de processamento p_j e está disponível para execução a partir de sua data de liberação r_j. A execução de uma tarefa j imediatamente após uma tarefa i requer um tempo de *setup* s_{ij}. Para a primeira tarefa da sequência, considera-se um tempo de configuração inicial s_{0j}. Os tempos de *setup* são não-antecipatórios, ou seja, o processo de preparação para a tarefa j só pode ser iniciado após sua liberação, em r_j.

O conjunto de tarefas J pode ser particionado em dois subconjuntos: J^-, contendo as tarefas que consomem recurso, e J^+, contendo aquelas que produzem recurso. O processamento de uma tarefa $j \in J^-$ reduz o nível do inventário em $\delta_j \leq 0$, enquanto o de uma tarefa $j \in J^+$ o aumenta em $\delta_j \geq 0$. Os recursos são armazenados em um inventário com capacidade limitada a I_C, que inicialmente já contém uma quantidade de recurso, definida como I_0. Um dos principais desafios do problema é garantir que, ao longo da sequência, o nível do inventário nunca ultrapasse sua capacidade máxima nem fique abaixo do necessário para executar tarefas consumidoras de recurso.

Uma solução do problema é representada por uma sequência $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, na qual cada tarefa $j \in J$ é executada exatamente uma vez. O tempo de conclusão C_j de uma tarefa j é definido da seguinte forma: se j for a primeira tarefa da sequência ($j = \pi_1$), então $C_j = r_j + s_{0j} + p_j$; caso contrário, considerando i a tarefa imediatamente anterior na sequência, tem-se $C_j = \max\{(C_i, r_j) + s_{ij} + p_j\}$.

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

O objetivo é determinar uma sequência π que minimize o tempo de conclusão da última tarefa, ou seja, o *makespan*, definido como $C_{\max} = \max_{j \in J}\{C_j\}$.

4. Formulações

4.1. Formulação indexada em posição

Esta formulação adapta o modelo originalmente proposto por Davari et al. [2020] para o $1|r_j, inv|C_{max}$. São definidas variáveis x_{js} que são ativadas caso a tarefa $j \in J$ seja processada na posição $s \in S$ da sequência, onde $S = \{1, ..., n\}$. O nível do inventário após o processamento de uma tarefa na posição $s \in S$ é representado pelas variáveis y_s. As variáveis C_s representam o tempo de conclusão da tarefa na posição s.

$$\min \quad C_n \tag{1}$$

Sujeito a:

$$\sum_{s=1}^{n} x_{js} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in J} x_{js} = 1 \qquad \forall s \in S \tag{3}$$

$$C_1 = \sum_{j \in J} (r_j + p_j + s_{0j}) x_{j1} \tag{4}$$

$$C_s \geq C_{s-1} + (p_j + s_{ij})(x_{is-1} + x_{js} - 1) \qquad \forall s \in S \setminus \{1\}, i \in J, j \in J \tag{5}$$

$$C_s \geq (r_j + p_j + s_{ij})(x_{is-1} + x_{js} - 1) \qquad \forall s \in S \setminus \{1\}, i \in J, j \in J \tag{6}$$

$$y_1 = I_0 + \sum_{j=1}^{n} \delta_j x_{j1} \tag{7}$$

$$y_s = y_{s-1} + \sum_{j=1}^{n} \delta_j x_{js} \qquad \forall s \in S \setminus \{1\} \tag{8}$$

$$C_s \geq 0 \qquad \forall s \in S \tag{9}$$

$$x_{js} \in \{0, 1\} \qquad \forall j \in J, s \in S \tag{10}$$

$$0 \leq y_s \leq I_C \qquad \forall s \in S \tag{11}$$

A função objetivo (1) minimiza o tempo de conclusão da tarefa na última posição da sequência. As restrições (2) e (3) estabelecem que cada tarefa deve ser alocada a uma posição apenas, e que uma posição só pode comportar uma única tarefa. O tempo de conclusão da primeira tarefa da sequência é computado pela restrição (4). As restrições (5) e (6) computam o tempo de término das demais tarefas. A restrição (7) computa o inventário após o processamento da primeira tarefa, enquanto as restrições (8) computam o inventário para as demais. As restrições (9) – (11) definem o domínio das variáveis.

4.2. Formulação indexada em arco

O modelo proposto é uma adaptação do apresentado por Velez-Gallego et al. [2016] para o $1|r_j, s_{ij}|C_{max}$, que utiliza uma formulação indexada em arcos, em que as variáveis de decisão x_{ij} assumem 1 caso a tarefa $j \in J'$ seja processada imediatamente após a tarefa $i \in J'$ na sequência. O conjunto J' é definido por $J' = J \cup \{0\}$, onde 0 é uma tarefa artificial utilizada para definir o início e o fim do sequenciamento. Variáveis b_i computam o tempo de início da tarefa $i \in J$, σ

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

é a variável contínua que computa o tempo ocioso total. Na adaptação para o problema proposto são introduzidas as variáveis y_{ij}, que computam o nível do inventário no arco $(i, j) : i, j \in J'$. A formulação pode ser expressa da seguinte forma:

$$\min \quad \beta + \sum_{i \in J'} \sum_{j \in J} s_{ij} x_{ij} + \sigma \tag{12}$$

Sujeito a:

$$\sum_{i \in J' \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in J' \tag{13}$$

$$\sum_{j \in J' \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in J' \tag{14}$$

$$b_0 = 0 \tag{15}$$

$$b_k + p_k + \sum_{j \in J' \setminus \{k\}} s_{jk} x_{jk} - \beta - \sum_{i \in J'} \sum_{j \in J' \setminus \{i\}} s_{ij} x_{ij} - \sigma \leq 0 \qquad \forall k \in J' \tag{16}$$

$$b_i + p_i + \sum_{k \in J' \setminus \{i\}} s_{ki} x_{ki} + M(x_{ij} - 1) - b_j \leq 0 \qquad \forall i \in J', j \in J \setminus \{i\} \tag{17}$$

$$r_j - b_j \leq 0 \qquad \forall j \in J' \tag{18}$$

$$\sum_{k \in J' \setminus \{j\}} y_{jk} - \sum_{i \in J' \setminus \{j\}} y_{ij} = \delta_j \qquad \forall j \in J \tag{19}$$

$$I_0 x_{0j} - y_{0j} = 0 \qquad \forall j \in J \tag{20}$$

$$\delta_i x_{ij} - y_{ij} \leq 0 \qquad \forall i, j \in J' \tag{21}$$

$$0 \leq y_{ij} \leq I_c \qquad \forall i, j \in J' \tag{22}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in J' \tag{23}$$

$$y_{ij} \geq 0 \qquad \forall i, j \in J' \tag{24}$$

$$b_j \geq 0 \qquad \forall j \in J' \tag{25}$$

$$\sigma \geq 0. \tag{26}$$

A Equação (12) contém a função objetivo, que é a minimização do *makespan*, composto pela soma de três termos: o tempo de processamento total $\beta = \sum_{j \in J} p_j$, os tempos de *setup* e o tempo ocioso total. As restrições (13) e (14) estabelecem que uma tarefa tem exatamente uma tarefa antecessora e uma tarefa sucessora. (15) define que uma tarefa *dummy* é processada no início do sequenciamento. As restrições (16) limitam o tempo de conclusão de uma tarefa à soma dos tempos de processamento, *setups* e tempo ocioso. As restrições (17) garantem que o início do processamento de uma tarefa só ocorre após o término da tarefa anterior. As restrições (18) garantem que as tarefas só serão iniciadas após a liberação. As Restrições (19) computam o nível do inventário nos arcos. (20) computa o inventário no arco inicial. As restrições (21) – (22) garantem que os limites do inventário são respeitados. Por fim, as restrições (23) – (26) definem o domínio das variáveis.

4.3. Formulação arco-inventário

Nesta formulação são utilizadas variáveis de decisão x_{ij}^q, que assumem 1 caso a tarefa $j \in J$ seja processada imediatamente após a tarefa $i \in J$ e o nível do inventário após o processamento é $q \in Q_j$. O conjunto Q_j é definido como $Q_j = \{q_j^{min}, ..., q_j^{max}\}$, com $q_j^{min} = 0$ e $q_j^{max} = I_c - \delta_j$

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

para $j \in J^-$, e $q_j^{min} = \delta_j$ e $q_j^{max} = I_c$ para $j \in J^+$. Essa definição garante que os limites do inventário serão respeitados pelo modelo.

As demais variáveis de decisão permanecem inalteradas em relação à formulação apresentada na Seção 4.2. A formulação encontra-se descrita a seguir:

$$\min \quad \beta + \sum_{i \in J'} \sum_{j \in J} \sum_{q=0}^{I_C} s_{ij} x_{ij}^q + \sigma \tag{27}$$

Sujeito a:

$$\sum_{i \in J' \setminus \{j\}} \sum_{q=0}^{I_C} x_{ij}^q = 1 \qquad \forall j \in J' \tag{28}$$

$$\sum_{j \in J' \setminus \{i\}} \sum_{q=0}^{I_C} x_{ij}^q = 1 \qquad \forall i \in J \tag{29}$$

$$\sum_{j \in J'} x_{0j}^{I_0+\delta_j} = 1 \tag{30}$$

$$\sum_{j \in J' \setminus \{i\}} x_{ji}^q - \sum_{j \in J' \setminus \{i\}} x_{ij}^{q+\delta_j} = 0 \qquad \forall i \in J, q \in Q_i \tag{31}$$

$$b_k + p_k + \sum_{j \in J' \setminus \{k\}} \sum_{q \in Q_k} s_{jk} x_{jk}^q - \beta - \sum_{i \in J'} \sum_{j \in J' \setminus \{i\}} \sum_{q \in Q_j} s_{ij} x_{ij}^q - \sigma \leq 0 \qquad \forall k \in J' \tag{32}$$

$$b_i + p_i + \sum_{k \in J' \setminus \{i\}} \sum_{q \in Q_i} s_{ki} x_{ki}^q + M \left(\sum_{q \in Q_j} (x_{ij}^q) - 1 \right) - b_j \leq 0 \qquad \forall i \in J, j \in J' \tag{33}$$

$$x_{ij}^q \in \{0, 1\} \qquad \forall i, j \in J', q \in Q_j \tag{34}$$

(15), (18), (25), (26).

No geral, a função objetivo e restrições assumem forma semelhante a formulação com variáveis de decisão indexadas somente em arco, com ligeiras modificações para acomodar as novas variáveis indexadas em arco-inventário. Destacam-se as restrições (30) que estabelecem o nível do inventário após a primeira tarefa da sequência, e (31), que determinam o nível do inventário nos demais arcos.

4.4. Formulação indexada em arco-tempo

A formulação, inspirada no trabalho de Pessoa et al. [2010], introduz as variáveis de decisão x_{ij}^t, onde $x_{ij}^t = 1$ se a tarefa $i \in J'$ acaba de ser processada e a tarefa $j \in J'$ se inicia no tempo $t \in \{0, 1, ..., t_{max} - p_j\}$. Caso contrário $x_{ij}^t = 0$. As variáveis contínuas y_t contabilizam o nível do inventário no instante $t \in \{0, 1, ..., t_{max}\}$, em que t_{max} é um limitante superior do *makespan*. O modelo proposto é descrito a seguir:

$$\min \quad C_{max} \tag{35}$$

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

Sujeito a:

$$\sum_{i \in J' \setminus \{j\}} \sum_{t=p_i}^{t_{max}-p_j} x_{ij}^t = 1 \qquad \forall j \in J' \qquad (36)$$

$$\sum_{j \in J} \sum_{t=s_{0j}+r_j}^{t_{max}-p_j} x_{0j}^t = 1 \qquad (37)$$

$$\sum_{\substack{j \in J' \setminus \{i\}, \\ t-p_j \geq 0}} x_{ji}^t - \sum_{\substack{j \in J' \setminus \{i\}, \\ t+p_i+p_j+s_{ij} \leq t_{max}}} x_{ij}^{max(t+p_i+s_{ij},\ r_j+s_{ij})} = 0 \qquad \forall i \in J, t \in T \qquad (38)$$

$$C_{max} \geq \sum_{i \in J \setminus \{j\}} \sum_{t=p_i}^{t_{max}-p_j} (t+p_j) x_{ij}^t \qquad \forall j \in J' \qquad (39)$$

$$y_0 = I_0 \qquad (40)$$

$$y_t = y_{t-1} + \sum_{i \in J'} \sum_{j \in J} \delta_j x_{ij}^t \qquad \forall t = \{1, ..., t_{max}\} \qquad (41)$$

$$0 \leq y_t \leq I_C \qquad \forall t \in \{0, ..., t_{max}\} \qquad (42)$$

$$C_{max} \geq 0 \qquad (43)$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall i \in J', j \in J', t \in T. \qquad (44)$$

A função objetivo (35) minimiza o *makespan*. As restrições (36) indicam que cada tarefa do conjunto deve ser processada uma vez. A restrição (37) define o início do processamento do *scheduling*, que deve ser feito em apenas uma máquina, após a data de liberação da primeira tarefa e o *setup* inicial. As restrições (38) garantem a conservação do fluxo, estabelecendo que se uma tarefa $i \in J'$ é finalizada no instante $t \in \{0, ..., t_{max}\}$, a próxima tarefa $j \in J'$ deve ser iniciada no instante $\max(t + p_i + s_{ij},\ \ r_j + s_{ij})$. As restrições (39) computam o tempo de término das tarefas. As restrições (40) e (41) computam o nível do inventário em cada instante t. As restrições (42) garantem que os níveis mínimo e máximo do inventário são respeitados. As restrições (43) e (44) definem a não negatividade dos tempos de término e o domínio das variáveis, respectivamente.

5. Experimentos Computacionais

As formulações apresentadas foram implementadas em linguagem C++ (g++ 13.2.0). Todos os experimentos foram executados em uma máquina com processador Intel Core i7-12700, 16 GB RAM e sistema operacional Ubuntu 24.04. Foi utilizado o resolvedor CPLEX 22.11 com tempo limite de 1 hora, limitado a 4 *threads* e com uso de memória da árvore limitado a 4 GB.

5.1. Geração de instâncias

A geração das instâncias baseou-se nos métodos descritos em Ovacikt e Uzsoy [1994] e Davari et al. [2020]. O conjunto é composto por cinco grupos de instâncias, variando em tamanho com $n \in \{10, 20, 30, 40, 50\}$. Os tempos de processamento p_j foram extraídos de uma distribuição uniforme discreta no intervalo $[1, \alpha]$, com $\alpha \in \{10, 100\}$. As datas de liberação foram geradas a partir de uma distribuição uniforme discreta no intervalo $[0, \tau \sum_{j \in J} p_j]$, onde $\tau \in \{0.2, 0.6, 1.0, 1.4, 1.8\}$. Os tempos de *setup* foram sorteados de uma distribuição uniforme no intervalo $[1, 200]$.

De forma análoga, os valores absolutos $|\delta_j|$ das variações de inventário foram obtidos de uma distribuição uniforme discreta entre 1 e 10. A capacidade do inventário I_c foi aleatoriamente definida no intervalo $[10\eta, 20\eta]$, com $\eta \in \{1, 3, 5\}$. Para cada combinação dos parâmetros α, τ e η foram geradas 3 instâncias, totalizando 450 instâncias no conjunto experimental.

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

5.2. Resultados agregados

A Tabela 1 resume os resultados agregados obtidos por cada formulação nos diferentes tamanhos de instâncias. Foram computados o número total de instâncias em que foi encontrada a solução ótima, o *gap* médio geral e o tempo médio de execução para os casos em que o modelo encontrou a solução ótima. Observa-se que o aumento no número de tarefas torna o problema significativamente mais difícil, especialmente na formulação baseada em posição. Embora essa abordagem tenha resolvido todas as instâncias com $n = 10$, este modelo apresentou grandes dificuldades nos demais cenários, com altos valores de *gaps* e baixa taxa de sucesso na obtenção de soluções ótimas.

Entre as formulações avaliadas, a baseada em arco foi a que apresentou o melhor desempenho, conseguindo resolver instâncias com até 50 tarefas. Além disso, para todos os valores de n, essa formulação encontrou o maior número de soluções ótimas e obteve tempos médios de resolução inferiores aos das demais abordagens.

5.3. Análise de sensibilidade

Para verificar como as características das instâncias influenciam na dificuldade de resolução do problema, foi realizada uma análise da taxa de resolução de cada formulação, segmentada entre os diferentes tamanhos de instância e valores dos parâmetros α, τ, e η, cujos resultados são apresentados nas Tabelas 2.

É possível notar que todas as formulações apresentam maior dificuldade para resolver instâncias onde $\alpha = 100$, visto que nesse cenário as tarefas tendem a apresentar maiores tempos de processamento e datas de liberação.

Especialmente no modelo arco-tempo, é observada uma queda na taxa de resolução conforme o valor de τ aumenta. Isso indica que tal formulação tem dificuldade em lidar com valores de datas de liberação maiores e mais espaçados, pois isso acarreta em maiores valores de *makespan* e, consequentemente, aumento no número de variáveis no modelo.

Tabela 1: Comparação de performance das diferentes formulações

Model	$n = 10$			$n = 20$			$n = 30$			$n = 40$			$n = 50$		
	#Opt	Gap (%)	Tempo (s)	#Opt	Gap (%)	Tempo (s)	#Opt	Gap (%)	Tempo (s)	#Opt	Gap (%)	Tempo (s)	#Opt	Gap (%)	Tempo (s)
Posição	90	0.00	6.38	1	1650	474.83	0	5822	–	0	5478	–	0	18128	–
Arco	90	0.00	1.19	33	3.56	259.76	22	2.80	212.90	8	2.44	582.29	1	2.40	196.46
Arco Inv	73	0.93	266.32	14	3.76	624.56	5	2.46	1355.72	1	2.23	1025.05	0	2.27	–
Arco Tempo	89	0.17	170.83	29	1.30	995.28	2	1.26	1426.42	0	1.47	–	0	7.83	–

6. Considerações Finais

Este trabalho apresentou o problema de sequenciamento de tarefas em uma máquina única, considerando datas de liberação, tempos de *setup* dependentes da sequência e restrições de inventário, com o objetivo de minimizar o *makespan*. Foram propostas quatro formulações distintas para modelar o problema: a primeira baseada em variáveis indexadas por posição; a segunda com variáveis de decisão indexadas por arcos; a terceira por arcos combinados com níveis de inventário; e a quarta por arcos e instantes de tempo.

Para a avaliação do desempenho dos modelos, foi gerado um conjunto composto por 450 instâncias, permitindo a análise do impacto de diferentes características das instâncias sobre a complexidade e capacidade de resolução do problema.

Entre as formulações propostas, o modelo indexado por arcos apresentou o melhor desempenho, encontrando a solução ótima em 154 instâncias, sendo o único capaz de resolver casos com até 50 tarefas.

Como direções para trabalhos futuros, destaca-se a ampliação dos experimentos para instâncias com maior número de tarefas ($n > 50$), o desenvolvimento de algoritmos exatos como

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

Tabela 2: Porcentagem de instâncias resolvidas para diferentes modelos e configurações dos parâmetros α, τ, η

(a) Modelo indexado em posição

Parâmetros	**n**				
	10	20	30	40	50
$\alpha = 10$	100%	2%	0%	0%	0%
$\alpha = 100$	100%	0%	0%	0%	0%
$\tau = 0.2$	100%	0%	0%	0%	0%
$\tau = 0.6$	100%	6%	0%	0%	0%
$\tau = 1.0$	100%	0%	0%	0%	0%
$\tau = 1.4$	100%	0%	0%	0%	0%
$\tau = 1.8$	100%	0%	0%	0%	0%
$\eta = 1$	100%	0%	0%	0%	0%
$\eta = 3$	100%	0%	0%	0%	0%
$\eta = 5$	100%	3%	0%	0%	0%

(b) Modelo indexado em arco

Parâmetros	**n**				
	10	20	30	40	50
$\alpha = 10$	100%	42%	38%	13%	2%
$\alpha = 100$	100%	31%	11%	4%	0%
$\tau = 0.2$	100%	6%	0%	0%	0%
$\tau = 0.6$	100%	17%	6%	0%	0%
$\tau = 1.0$	100%	44%	17%	6%	0%
$\tau = 1.4$	100%	50%	50%	11%	6%
$\tau = 1.8$	100%	67%	50%	28%	0%
$\eta = 1$	100%	30%	23%	10%	0%
$\eta = 3$	100%	30%	27%	7%	3%
$\eta = 5$	100%	50%	23%	10%	0%

(c) Modelo arco-inventário

Parâmetros	**n**				
	10	20	30	40	50
$\alpha = 10$	89%	18%	7%	2%	0%
$\alpha = 100$	73%	13%	4%	0%	0%
$\tau = 0.2$	100%	6%	0%	0%	0%
$\tau = 0.6$	61%	0%	0%	0%	0%
$\tau = 1.0$	83%	17%	0%	0%	0%
$\tau = 1.4$	78%	22%	6%	0%	0%
$\tau = 1.8$	83%	33%	22%	6%	0%
$\eta = 1$	100%	30%	10%	3%	0%
$\eta = 3$	80%	7%	0%	0%	0%
$\eta = 5$	63%	10%	7%	0%	0%

(d) Modelo arco-tempo

Parâmetros	**n**				
	10	20	30	40	50
$\alpha = 10$	100%	40%	4%	0%	0%
$\alpha = 100$	98%	24%	0%	0%	0%
$\tau = 0.2$	100%	72%	11%	0%	0%
$\tau = 0.6$	100%	50%	0%	0%	0%
$\tau = 1.0$	100%	22%	0%	0%	0%
$\tau = 1.4$	100%	17%	0%	0%	0%
$\tau = 1.8$	94%	0%	0%	0%	0%
$\eta = 1$	97%	3%	0%	0%	0%
$\eta = 3$	100%	50%	0%	0%	0%
$\eta = 5$	100%	43%	7%	0%	0%

branch-and-cut ou *branch-cut-and-price*, além da investigação de abordagens heurísticas para obtenção de melhores *bounds*.

Referências

Allahverdi, A., Ng, C. T., Cheng, T. C. E., e Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187:985–1032.

Briskorn, D., Choi, B.-C., Lee, K., Leung, J., e Pinedo, M. (2010). Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, 207(2):605–619.

Briskorn, D. e Leung, J. Y. (2013). Minimizing maximum lateness of jobs in inventory constrained scheduling. *Journal of the Operational Research Society*, 64(12):1851–1864.

Briskorn, D. e Pesch, E. (2013). Variable very large neighbourhood algorithms for truck sequencing at transshipment terminals. *International Journal of Production Research*, 51(23-24):7140–7155.

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

Carlier, J. e Kan, A. R. (1982). Scheduling subject to nonrenewable-resource constraints. *Operations Research Letters*, 1(2):52–55.

Davari, M., Ranjbar, M., De Causmaecker, P., e Leus, R. (2020). Minimizing makespan on a single machine with release dates and inventory constraints. *European Journal of Operational Research*, 286(1):115–128.

Fan, J., Öztop, H., Tasgetiren, M., e Gao, L. (2019). A variable block insertion heuristic for single machine with release dates and sequence dependent setup times for makespan minimization. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, p. 1676–1683, Xiamen, China. IEEE.

Ghorbanzadeh, M., Ranjbar, M., e Jamili, N. (2019). Transshipment scheduling at a single station with release date and inventory constraints. *Journal of Industrial and Production Engineering*, 36(5):301–312. URL https://doi.org/10.1080/21681015.2019.1647300.

Graham, R., Lawler, E., Lenstra, J., e Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E. J. e Korte, B., editors, *Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*, volume 5 of *Annals of Discrete Mathematics*, p. 287 – 326. Elsevier.

Guerra, F., Morais, R., Moura, A. C., e Subramanian, A. (2024). Uma heurística populacional híbrida para problemas de sequenciamento de tarefas com datas de liberação e restrições de inventário. In *Anais do LVI Simpósio Brasileiro de Pesquisa Operacional*, p. Anais Eletrônicos, Fortaleza. Galoá.

Montoya-Torres, J., González-Solano, F., e Soto-Ferrari, M. (2012). Deterministic machine scheduling with release times and sequence-dependent setups using random-insertion heuristics. *International Journal of Advanced Operations Management*, 4(1/2):4–26.

Morais, R., Bulhões, T., e Subramanian, A. (2024). Exact and heuristic algorithms for minimizing the makespan on a single machine scheduling problem with sequence-dependent setup times and release dates. *European Journal of Operational Research*, 315(2):442–453.

Ovacikt, I. e Uzsoy, R. (1994). Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 32(6):1243–1263.

Pessoa, A., Uchoa, E., de Aragão, M. P., e Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2:259–290.

Pinedo, M. L. (2022). *Scheduling: Theory, Algorithms, and Systems*. Springer-Verlag, Nova Iorque, 6 edition.

Slowiński, R. (1984). Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. *European Journal of Operational Research*, 15(3):366–373.

Toker, A., Kondakci, S., e Erkip, N. (1991). Scheduling under a non-renewable resource constraint. *Journal of the Operational Research Society*, 42(9):811–814.

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br

Velez-Gallego, M. C., Maya, J., e Montoya-Torres, J. (2016). A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan. *Computers & Operations Research*, 73:132–140.

DOI: 10.59254/sbpo-2025-212140
https://proceedings.science/p/212140?lang=pt-br