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

Framework BRKGA-VNS para sequenciamento na etapa de têmpera em uma siderurgia

João Vitor Cordeiro Sánchez
Universidade Federal de Minas Gerais
Belo Horizonte, MG - Brasil
jvcsanchez96@gmail.com

Daniel Nascimento Paiva
Universidade Federal de Minas Gerais
Belo Horizonte, MG - Brasil
daniel-np@ufmg.br

Anderson Alves Tzitas
Vallourec
Belo Horizonte, MG - Brasil
anderson.atzitas@vallourec.com

RESUMO

Este artigo propõe um *framework* híbrido baseado no *Biased Random-Key Genetic Algorithm* (BRKGA) acoplado a um *General Variable Neighborhood Search* (GVNS) para o problema de sequenciamento de máquina única com tempos de setup dependentes da sequência com janela de tempo. Motivado pelo ganho de performance e capacidade de um dos processos críticos da siderurgia, o processo de têmpera, o modelo inclui uma condição de Dupla Têmpera, onde pares de *jobs* não podem ser feitos diretamente um após o outro. Experimentos computacionais utilizando oito instâncias reais indicam que o *framework* supera o modelo de programação linear e a heurística VNS anterior, obtendo *gaps* melhores em média de 21,3 % no *makespan* em 15 minutos de execução quando comparados a 60 minutos de duas formulações inteiras em um solver comercial. Os resultados confirmam o potencial da abordagem para aplicações reais de pesquisa operacional e meta-heurísticas na indústria mesmo em projetos industriais com regras de negócio específicas.

PALAVRAS CHAVE. Sequenciamento de uma máquina, BRKGA, GVNS, Meta-heurísticas, Janelas de tempo, Dupla Têmpera.

Tópicos: POI – PO na Indústria, Meta-heurísticas.

ABSTRACT

This paper proposes a hybrid framework based on the *Biased Random-Key Genetic Algorithm* (BRKGA) coupled with a *General Variable Neighborhood Search* (GVNS) for the single-machine scheduling problem with sequence-dependent setup times and time windows. Motivated by performance and capacity gains in the steelmaking heat-treatment stage, the model includes a *Double Quenching* rule, where certain pairs of *jobs* cannot be processed consecutively. Computational experiments on eight real instances indicate that the framework outperforms the linear programming model and the previous VNS heuristic, achieving average gap improvements of **21.3%** within **15 minutes** of execution, compared with **60 minutes** for **two complete integer programming (MIP) formulations** in a commercial solver. The results confirm the potential of the approach for real applications of **operations research** and metaheuristics in industry, even in operational-level projects with specific business rules.

KEYWORDS. Single-machine Scheduling, BRKGA, GVNS, Metaheuristics, Time Windows, Double Quenching.

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

1. Introdução

O crescimento de dados operacionais disponíveis — provenientes de sensores industriais, sistemas de automação da manufatura e plataformas de gestão integrada — aumentou a necessidade de decisões rápidas e confiáveis em todos os elos das cadeias produtivas. A Pesquisa Operacional converte essa abundância de informação em estratégias tangíveis para alocação de recursos, roteirização e planejamento de produção. Quando seus resultados teóricos são levados ao chão de fábrica, transformam-se em ganhos mensuráveis de capacidade, redução de custos e mitigação de desperdícios — benefícios cada vez mais relevantes diante de metas de desempenho cada vez mais rigorosas.

Nesse contexto, meta-heurísticas, tanto de solução única quanto populacionais, passaram a ser ferramentas cruciais para preencher a lacuna entre modelos de programação linear inteira e prazos reais de decisão, culminando em abordagens híbridas que equilibram exploração e intensificação. O desafio moderno não é apenas encontrar soluções factíveis, mas fazê-lo dentro de janelas temporais compatíveis com a volatilidade de um ambiente industrial, restrições de capacidades e flutuações de demanda. Aplicar conceitos clássicos — relaxações lagrangeanas, dualidade, teoria dos grafos — demonstra que a utilidade social da matemática depende de sua operacionalização. Este trabalho mostra como a integração inteligente de algoritmos transforma dados abundantes em ganhos concretos de eficiência industrial.

A etapa de têmpera em questão requer o aquecimento sequencial de ordens em um forno de uma única linha. Cada lote, *ordem*, produz peças de dimensões distintas, que possuem tempos de processamento diferentes, e exigem tempos de preparação (*setups*) dependentes da sequência. Além disso, restrições de disponibilidade de cada ordem (*release dates*), prazos de entrega (*due dates*) e paradas obrigatórias de manutenção aumentam a complexidade operacional. Na prática, programadores humanos recorrem a planilhas heurísticas que consomem horas e raramente garantem boa qualidade de solução.

Nos últimos anos, meta-heurísticas tornaram-se a alternativa mais viável para problemas de sequenciamento industriais de médio porte, onde formulações exatas perdem escalabilidade [Nogueira et al., 2019]. Em 2024, Sá et al. [2024] desenvolveram um *Variable Neighborhood Search* (VNS) para o mesmo cenário industrial utilizando sua resposta como *warmstart* para modelos de programação linear e reduzindo ainda mais o *makespan*. Entretanto, a geração da solução inicial para o VNS por heurísticas gulosas limitou o potencial da busca local e na época não foram consideradas as janelas de tempo.

Neste trabalho, introduzimos um *framework* híbrido em duas fases:

1. Um BRKGA modificado [Gonçalves e Resende, 2011] que gera populações diversificadas de sequências, evitando as inviáveis;

2. A melhor sequência alimenta um GVNS [Mladenović e Hansen, 1997] que emprega perturbações graduais e busca local *Variable Neighborhood Descent* (VND). O BRKGA explora o espaço de soluções, enquanto o GVNS intensifica a busca em regiões promissoras.

Adicionalmente, incorporamos uma nova regra de dupla têmpera, mais aderente ao processo produtivo. Os parâmetros das meta-heurísticas foram calibrados utilizando o Optuna, uma biblioteca que através de inferências experimentais aproxima os valores ótimos de cada hiperparâmetro.

Embora BRKGA e GVNS já tenham apresentado bons resultados isoladamente, combinou-se ambas as técnicas para tratar, simultaneamente, *release dates*, *due dates*, *setups* dependentes da sequência e a nova regra de dupla têmpera. Preenchemos essa lacuna com um arranjo híbrido em que o BRKGA fornece diversidade genética e robustez a inviabilidades, enquanto o GVNS garante

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

intensificação sistemática sem necessidade de parâmetros de vizinhança estáticos. Esse desenho exige pouco ajuste manual e se mostra escalável a instâncias com mais de 40 *jobs*.

Aparentemente o aumento da complexidade da instância passa a ser menos atrelado a quantidade de *jobs* e mais ao tamanho das janelas de tempo e bloqueios provenientes da regra de dupla têmpera. Ensaios preliminares revelam reduções médias de 21,3 % em oito instâncias industriais, dentro de um tempo-limite de 15 min por execução, contabilizando BRKGA e VNS. Esses resultados não podem ser comparados diretamente a Sá et al. [2024] por conta das mudanças de regra e adição de janela de tempo em todos os casos testados.

2. Referencial Teórico

Nogueira *et al.* comparam quatro modelos inteiros clássicos (CTP, APD, TI e ATI) para o *Single-Machine Scheduling Problem* (SMSP) com tempos de preparação dependentes da sequência e *release dates*. Embora as formulações indexadas pelo tempo (TI e ATI) apresentem relaxações lineares mais apertadas, a explosão no número de variáveis inviabiliza sua aplicação quando o horizonte alcança semanas. No mesmo contexto, Keha et al. [2009] apontam desempenho superior de modelos baseados em posições quando o tempo precisa ser discretizado em minutos, reforçando a busca por abordagens heurísticas em instâncias reais de larga escala.

Entre as meta-heurísticas, o VNS Mladenović e Hansen [1997] se destaca pela capacidade de escapar de ótimos locais por meio de perturbações sistemáticas de vizinhança. A versão *GVNS* proposta em Hansen et al. [2006] adiciona uma fase de busca local VND ao final de cada perturbação, intensificando a exploração em regiões promissoras. Nogueira et al. [2022] integraram VNS com relaxação Lagrangeana para o SMSP com *release* e *due dates*, obtendo soluções de alta qualidade em horizontes de curto e longo prazo. Rosa et al. [2018] avaliaram variações de VNS no SMSP com penalidades *earliness/tardiness* e confirmaram sua robustez. Para problemas com janelas de tempo, Ye et al. [2024] aplicaram GVNS ao TSP-TW e mostraram que a técnica compete com heurísticas do estado da arte em múltiplos objetivos.

O BRKGA foi proposto por Gonçalves e Resende [2011] como evolução dos *random-keys* originais de Bean Bean [1994]. Cada descendente herda o gene do pai elite com probabilidade ρ_e, mantendo diversidade com uma fração de mutantes. A publicação original defende configurações de P_e entre 0,10 e 0,25, P_m entre 0,10 e 0,20, probabilidade de herança ρ_e entre 0,50 e 0,80 e população próxima de an indivíduos. A facilidade de inserir penalidades no decodificador torna o BRKGA viável para restrições específicas, como as subsequências bloqueadas da dupla têmpera. Ma et al. [2018] demonstraram que populações diversificadas do BRKGA servem de ponto de partida eficaz para intensificação por vizinhanças, combinando as duas técnicas em problemas de lotes paralelos com deterioração.

No ambiente da têmpera siderúrgica, trabalhos recentes de Tzitas et al. [2024] adaptaram formulações exatas, enquanto Sá et al. [2024] mostraram que o VNS, aliado a uma heurística construtiva, reduz significativamente o *makespan*. A evolução natural é integrar a exploração global do BRKGA à intensificação do GVNS, mantendo tempos de execução compatíveis com a tomada de decisão diária — proposta central deste artigo, em que as regras de negócio são todas consideradas num *framework* híbrido e ágil.

3. Modelagem e Framework Meta-heurístico

Este trabalho estende as formulações apresentadas FA1 e FA2 em Tzitas et al. [2024] incorporando:

1. Datas de entrega *(due dates)*;

2. Revisão da regra de dupla têmpera para os *jobs* que não devem vir agrupados;

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

3. Um *framework* BRKGA-GVNS;

4. População adaptativa $\alpha \cdot |J|$;

5. Otimização dos hiperparâmetros com Optuna.

A seguir descrevemos cada componente.

Tabela 1: Notação dos conjuntos, variáveis e parâmetros

Símbolo	**Descrição**
Conjuntos	
J	Conjunto de ordens (*jobs*)
B	Conjunto de pares bloqueados (u, v) da regra de Dupla Têmpera
T	Conjunto de índices de posição/tempo discretizado
A	Conjunto de arcos (i, j) na formulação baseada em grafos
Parâmetros	
n	Número total de operações $(\|J\| + \|J'\|)$
p_j	Tempo de processamento da ordem j
s_{ij}	Tempo de preparação ao trocar de i para j
r_j	*Release date* da ordem j
d_j	*Due date* da ordem j
τ_j	Intervalo mínimo entre duas ordens de dupla têmpera associadas a j (Eq. 3)
Variáveis de decisão	
C_j	Tempo de conclusão da ordem j
C_t	Tempo de conclusão da ordem alocada à posição t (FA2)
x_{ij}^t	Variável binária: 1 se i precede j na posição t (FA2)
y_i^t	Variável binária: 1 se a ordem i ocupa a posição t (FA2)
Meta-heurísticas	
u_j	Chave aleatória associada à ordem j no BRKGA
p_e	Proporção de indivíduos elite no BRKGA
p_m	Proporção de mutantes no BRKGA
ρ_e	Probabilidade de herança da elite no cruzamento
α	Fator populacional: $\|P\| = \alpha \|J\|$

3.1. Extensões de Modelagem

3.1.1. Inserção de *due date* na Modelagem

Para a FA1, a inserção de *due date* é representada pela equação:

$$C_j \leq d_j \quad \forall\, j \in J' \tag{1}$$

Para a FA2, a inserção de *due date* é representada pela equação:

$$\sum_{i \in J} \sum_{j \in J} d_j x_{ij}^{t-1} \geq C_t \quad \forall t = 1...n \tag{2}$$

As expressões (1) e (2) definem que o tempo de completar uma ordem não deve ultrapassar a *due date*.

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

3.2. Regra de Dupla Têmpera

Como discutido por Tzitas et al. [2024], determinados *jobs* apresentam relações de dependência que exigem tratamento específico no modelo de sequenciamento. Essas relações podem incluir a necessidade de que dois *jobs* sejam processados em seguida, ou, no caso de *jobs* de bloqueio, que não sejam executados em sequência direta. No entanto, com a evolução das exigências do ambiente produtivo, tornou-se necessário refinar matematicamente a maneira que essa regra de negócio é modelada, visando maior aderência ao processo.

Dessa forma, a regra de bloqueio foi revisada e adaptada para refletir essa nova realidade operacional. Para representar essa restrição temporal, foi introduzido o parâmetro τ_j, que define o intervalo mínimo de tempo que deve ser respeitado entre duas ordens de dupla têmpera associadas a um *job* específico j, de modo a garantir a viabilidade da sequência produtiva. A formalização dessa restrição é apresentada na equação (3).

$$\tau_j = \frac{p_j}{pieces_j} \times \sum_k \text{Slot}_k \times 2 \quad \forall j \in J \tag{3}$$

Onde:

- p_j: tempo de processamento do *job* j;
- pieces_j: número de peças do *job* j;
- Slot_k: capacidade de cada slot disponível (posição onde as peças serão processadas).

Para a FA1, a regra de dupla têmpera é representada pela equação:

$$C_v \geq C_u + \tau_u + p_v \;\; \forall (u,v) \in B \tag{4}$$

Para a FA2, a regra de dupla têmpera é representada pelas equações:

$$y_i^t = \sum_{j \in J} x_{ij}^t \tag{5}$$

$$\sum_{t'>t} y_v^{t'} \geq y_u^t - \sum_{t'>t} y_u^{t'} \quad \forall (u,v) \in B, \forall t \in T \tag{6}$$

$$\sum_{t_1 \leq t < t_2} \sum_{(i,j) \in A} (s_{ij} + p_j) x_{ij}^t \geq \tau_u (y_u^{t_1} - y_v^{t_2} - 1) \quad \forall (u,v) \in B, \forall t_1, t_2 \in T \times T : t_1 > t_2 \tag{7}$$

3.3. Framework BRKGA-GVNS

O fluxo completo é mostrado na Figura 1. O algoritmo inicia com um BRKGA gerando soluções-semente; o melhor cromossomo decodificado alimenta um GVNS, que aplica perturbações *shaking* progressivas e uma busca local VND. Caso exista uma solução exata superior (solver MIP), esta é armazenada para análise de *gap* de otimalidade.

O *framework* meta-heurístico busca também minimizar o *makespan*, mas esse agora penalizado por cada violação de restrição de janela de tempo e dupla têmpera.



Figura 1: Fluxo do *framework* proposto.

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

3.4. BRKGA com Decodificação Heurística Guiada

Neste trabalho, propomos um aprimoramento do BRKGA, originalmente descrito por Gonçalves e Resende [2011], para o problema de sequenciamento de máquina única. O algoritmo foi adaptado de forma a incorporar regras heurísticas diretamente no processo de decodificação, de modo a evitar a geração de soluções inviáveis e reduzir penalizações excessivas em instâncias com forte restrição temporal. Cada cromossomo é representado por um vetor de chaves aleatórias $u \in [0,1]^{|J|}$, em que cada valor está associado a uma ordem $j \in J$. A ordenação dessas chaves define a sequência de produção.

Na sua forma canônica, o BRKGA é agnóstico a regras específicas do domínio — como janelas de tempo rígidas ou pares de ordens proibidas. Nossa contribuição reside também na especialização desse *framework* para o contexto industrial estudado, por meio de dois mecanismos centrais incorporados ao decodificador: enviesamento temporal e filtragem por bloqueios.

O enviesamento temporal é proporcional ao intervalo de tempo viável de cada ordem, este é uma inovação relevante no processo de geração da população inicial. Para isso, mapeamos o intervalo entre a data de liberação r_j e a data de entrega d_j de cada ordem para o espaço $[0,1]$, conforme:

$$u_j \sim \mathcal{U}\left[\max\left(0.05, \frac{r_j - r_{\min}}{d_{\max} - r_{\min}}\right), \min\left(0.95, \frac{d_j - r_{\min}}{d_{\max} - r_{\min}}\right)\right] \tag{8}$$

Essa abordagem garante que ordens com janelas mais restritas sejam associadas a chaves aleatórias mais concentradas dentro de seu intervalo viável, por exemplo, admita um horizonte global de 30 dias, contado de $t = 0$ (1º de janeiro) até $t = 30$ (30 de janeiro). Seja $[r_j, d_j] = [10, 20]$ a janela de uma ordem que só pode iniciar no dia 10 e deve terminar, no máximo, no dia 20.

A regra de enviesamento mapeia qualquer data $t \in [0, 30]$ para a faixa unitária $[0, 1]$ por meio da transformação afim:

$$\phi(t) = \frac{t - r_{\min}}{d_{\max} - r_{\min}} = \frac{t - 0}{30 - 0} = \frac{t}{30}.$$

Aplicando-a às extremidades da janela da ordem, obtém-se:

$$a_j = \phi(10) = \frac{10}{30} \approx 0{,}33, \qquad b_j = \phi(20) = \frac{20}{30} \approx 0{,}66.$$

Portanto a chave aleatória associada a essa ordem é sorteada em:

$$u_j \sim \mathcal{U}[0{,}33,\ 0{,}66],$$

Isto promove ordenações iniciais menos propensas à inviabilidade. Além disso, os indivíduos gerados como mutantes continuam sendo criados com chaves aleatórias puramente uniformes em $[0, 1]$, garantindo a exploração global do espaço de soluções.

Além da janela de tempo, o problema estudado impõe uma forte restrição: certos pares de ordens não podem ser executados de forma consecutiva, em nenhuma hipótese. Esses pares bloqueados, definidos como $(j_i, j_2) \in B$, representam situações de dupla têmpera ou interferência industrial que devem ser evitadas.

Para isso, durante a decodificação, propõe-se um algoritmo construtivo, chamado de filtragem por bloqueios, que percorre as chaves ordenadas e monta a sequência π final de forma incremental. A cada iteração, o *job* com menor chave restante é avaliado quanto à sua posição em relação à última ordem alocada:

1. Se a última ordem alocada for j_k, e o *job* atual j_i formar um par proibido com ela $(j_k, j_i) \in B$, esse *job* é temporariamente postergado e armazenado em uma fila de espera.

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

2. O processo continua com o próximo job, e a fila de espera é reavaliada quando novas posições livres forem identificadas.
3. Ao final, se algum job da fila não puder ser posicionado sem violar os bloqueios, ele é inserido com um *gap* proposital, forçando espaçamento via penalidade na função objetivo.

Essa estratégia impede que a penalização da função objetivo se torne dominante nas gerações iniciais, favorecendo a herança geracional de soluções viáveis.

Algorithm 1: Decodificador Sequencial com Filtragem de Bloqueios

Input: Chaves aleatórias u, conjunto de jobs J, pares bloqueados B
Output: Sequência viável π
Ordenar J conforme u_j crescente; Inicializar sequência $\pi \leftarrow [,]$; Inicializar fila de espera $F \leftarrow [,]$; **foreach** *job* j *em* J *ordenado* **do**
  if π *está vazia* **then**
    $\pi \leftarrow \pi + [j]$;
  end
  else if $(\pi[-1], j) \in B$ **then**
    $F \leftarrow F + [j]$; `// job proibido de vir após anterior`
  end
  else
    $\pi \leftarrow \pi + [j]$;
  end
end
while $F \neq \emptyset$ **do**
  Retirar j de F e inserir ao final de π com espaçamento;
end
return π

As duas modificações descritas não alteram a estrutura canônica do BRKGA, mantendo a essência do algoritmo. No entanto, atuam onde o BRKGA clássico é mais neutro: na geração inicial e na interpretação das soluções. Essa abordagem é mais eficaz em problemas onde existem múltiplas restrições difíceis de serem satisfeitas por amostragem aleatória e deseja-se manter a leveza computacional e o populacionismo inerente ao BRKGA original.

O resultado é um algoritmo que combina a simplicidade do BRKGA original com especializações heurísticas guiadas por domínio, com ganhos objetivos de desempenho em tempo e viabilidade das soluções desde as primeiras gerações, sem eliminar a capacidade do algoritmo de explorar qualquer tipo de solução através dos indivíduos mutagênicos.

O fluxo final do algoritmo utilizado está apresentado no Algoritmo 2:

3.5. Calibração de Hiperparâmetros com Optuna

Para garantir a melhor performance do BRKGA, realizamos uma calibração automatizada dos hiperparâmetros utilizando o *framework* Optuna Akiba et al. [2019]. Foram otimizados os seguintes parâmetros:

- α: fator multiplicativo da população (popSize $= \alpha \cdot |J|$);
- p_e: fração de indivíduos considerados elites;
- p_m: proporção de mutantes;
- ρ_e: probabilidade de herança da elite no crossover.

Cada *trial* executou o BRKGA modificado por 100 segundos, e o valor retornado foi o *makespan* da melhor solução encontrada. Utilizou-se um passo de 10% no intervalo de busca de cada hi-

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

Algorithm 2: BRKGA com Decodificação Heurística

Entrada: Instância (J, r_j, d_j, B, S)
Saída : Melhor sequência π^*
Inicializar população P com chaves enviesadas conforme $[r_j, d_j]$;
repeat
  Avaliar P com função objetivo penalizada;
  Separar P em elites E e não-elites N;
  foreach *par* $(e, n) \in E \times N$ **do**
    $c \leftarrow$ Cruzamento(e, n, p);
    Decodificar c com heurística de inserção com bloqueio;
    Adicionar c à nova população P';
  Adicionar mutantes à P' com chaves $\sim \mathcal{U}(0, 1)$;
  Atualizar população $P \leftarrow E \cup P'$;
until *Critério de parada*;
return $\pi^* = \arg\min_{\pi \in P} FO(\pi)$

perparâmetro, dentro dos intervalos originalmente sugeridos por Gonçalves e Resende [2011]. Os melhores valores encontrados foram adotados para todas as instâncias posteriores, por apresentarem desempenho robusto e estável.

Essa abordagem representa um casamento elegante entre técnicas modernas de aprendizado (Optuna) e meta-heurísticas, promovendo um balanceamento entre exploração e intensificação e aumentando significativamente a qualidade das soluções e a adaptabilidade do algoritmo às diferentes instâncias do problema.

3.5.1. General Variable Neighborhood Search (GVNS)

O **Algoritmo 3** mostra o funcionamento do GVNS. Foi utilizado o total de 5 vizinhanças de *shaking* $l = 1, \ldots, L_{\max}$, onde $L_{\max} = 5$.

Algorithm 3: General Variable Neighborhood Search (GVNS)

Entrada: X_0: **Solução inicial**; L_{max}: **número máximo de vizinhanças de *shaking***.
Saída: X: **melhor solução**
$X \leftarrow X_0$
$l \leftarrow 1$
while $l \leq L_{max}$ **do**
  $X' \leftarrow$ *Shaking*(X, l)
  $X'' \leftarrow$ *VND*(X')
  if *Makespan*(X'') $<$ *Makespan*(X) **then**
    $X \leftarrow X''$
    $l \leftarrow 1$
  end
  else
    $l \leftarrow l + 1$
  end
end
return X

As 5 vizinhanças de *shaking* estão listadas na Tabela 2:

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

Tabela 2: Vizinhanças de *shaking* utilizadas no GVNS

Vizinhança	**Funcionamento**
Inserção	Seleciona um *job* aleatório e o insere em uma posição aleatória da sequência
Inserir Dupla Têmpera	Seleciona um *job* dentre os pares de dupla têmpera e o insere em uma posição aleatória
2-opt	Seleciona 2 *jobs* aleatórios e inverte o segmento entre eles
3-opt	Seleciona 3 *jobs* aleatórios e aplica uma das reorganizações possíveis entre os três
Embaralhar segmento	Seleciona 2 *jobs* aleatórios e embaralha o segmento entre eles

3.5.2. VND

O funcionamento do VND está apresentado no Algoritmo 4. Ele percorre sequencialmente as vizinhanças de busca local k-*swap*, variando $k = 2$ até $k = 4$. As buscas locais para 2-*swap* consideram o conceito de *first-improvement* (Primeira Melhora), já as vizinhanças 3-*swap* e 4-*swap* consideram o conceito de *best-improvement* (Melhor Melhora).

Algorithm 4: Variable Neighborhood Descent (VND)

Entrada: X_0: Solução inicial do GVNS; K_{max}: Número máximo de vizinhanças de busca local
Saída: X: Melhor solução encontrada

$X \leftarrow X_0$
$k \leftarrow 2$
while $k \leq K_{max}$ **do**
  $X_{\text{new}} \leftarrow$ *K-Swap Local Search*(X, k)
  if *Makespan*(X_{new}) *¡ Makespan*(X) **then**
    $X \leftarrow X_{\text{new}}$
    $k \leftarrow 2$
  end
  else
    $k \leftarrow k + 1$
  end
end
return X

4. Resultados Computacionais

Os algoritmos foram implementados em *Python 3.12* utilizando o *IDE Visual Studio Code*. Os testes empregaram oito cronogramas reais de produção, disponibilizados pelos responsáveis pelo sequenciamento da têmpera, com horizontes que variam de 20 dias a 6 meses. Todas as execuções foram realizadas em um *notebook* com *Intel Core i7-1270P @ 2.2 GHz e 32 GB de RAM*. O limite de tempo foi fixado em 600 seg. para o BRKGA e 300 seg. adicionais para o GVNS; os modelos de programação linear tiveram 3600 seg. Apesar desse limite superior, o BRKGA convergiu, em média, aos seus melhores cromossomos em 68 seg., nunca excedendo 90 seg.

A aplicação do *framework* demonstrou-se eficaz a partir dos ganhos de *gaps*. Os resultados obtidos foram satisfatórios, especialmente ao integrar as duas meta-heurísticas. Em três instâncias (B_42_3, C_34_8, E_30_2), os *gaps* de otimalidade da FA2 foram 57,6%, 72,0% e 48,8%

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

Tabela 3: Comparação de *% gap* relativo ao melhor *lower bound*

Instância	***jobs***	**Dupla-Têmperas**	**Meta-heurísticas**		**Formulações**	
			BRKGA	**BRKGA–GVNS**	**FA1**	**FA2**
A_23_2	23	2	0.0%	0.0%	7.9%	0.0%
B_42_3	42	3	5.4%	5.1%	Infeasible	57.6%
C_34_8	34	8	15.3%	14.9%	25.5%	72.0%
D_46_1	46	1	0.0%	0.0%	87.5%	0.0%
E_30_2	30	2	1.1%	1.1%	2.1%	48.8%
F_32_5	32	5	4.1%	4.1%	Infeasible	8.8%
G_24_7	24	7	20.9%	11.3%	12.0%	18.0%
H_28_6	28	6	8.1%	8.1%	Infeasible	10.1%
Média	–	–	6.9%	**5.7%**	27.0%	26.9%

respectivamente, após a aplicação do *framework* foi possível encontrar gaps com melhora média de 52,4%.

A Tabela 3 mostra o *gap* percentual frente ao melhor *lower bound* disponível para cada instância. O *framework* BRKGA–GVNS superou ambas as formulações exatas (FA1 e FA2) em todas as oito instâncias, ao mesmo tempo em que o BRKGA puro já demonstrou ganhos expressivos. Nas instâncias B, F e H o ganho médio sobre o FA1 é incalculável, dado que não fora possível encontrar resposta viável. Além disso, o GVNS refinou a solução do BRKGA em 9,6% (instância G), com um ganho médio de 1,3%.

O tempo de resolução dos modelos inteiros alcançou o limite de uma hora em 75% dos casos, enquanto o novo *framework* concluiu em menos de 10 min, apresentando vantagens práticas evidentes para uso em ambiente produtivo em tempo satisfatório.

5. Conclusão

Este trabalho teve como meta reduzir o tempo de resolução de instâncias reais de sequenciamento da etapa de têmpera em uma indústria siderúrgica, problema caracterizado por janelas de tempo, tempos de preparação dependentes da sequência e pela regra de dupla têmpera. Para isso, propusemos um *framework* híbrido que combina a exploração diversificada do BRKGA com a intensificação de um GVNS, cujos hiperparâmetros são otimizados com auxílio da biblioteca *Optuna*.

Os experimentos conduzidos em oito cronogramas reais mostraram que o BRKGA–GVNS atingiu *gaps* médios de apenas 5,7 % em relação ao melhor *lower bound* disponível, superando as duas formulações exatas em todas as instâncias e obtendo soluções em menos de 10 min. Apesar do limite de execução de 600 seg. para a fase genética, o BRKGA convergiu a seus melhores cromossomos, em média, em 68 seg. — nunca excedendo 90 seg. —, fornecendo pontos de partida de alta qualidade para o GVNS. Os ganhos observados ratificam o potencial prático da proposta: a ferramenta pode ser executada várias vezes ao dia, permitindo reotimizações rápidas sempre que surgirem ordens urgentes ou indisponibilidades de equipamento na fábrica.

Como desdobramentos futuros, pretende-se:

1. Estender o modelo para múltiplos fornos operando em paralelo, o que aumentará o grau de interdependência entre decisões de sequenciamento e carregamento;

2. Avaliar a robustez do algoritmo diante da incerteza nos tempos de preparação, incorporando cenários e distribuições estocásticas de segurança na função-objetivo; e

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

3. Integrar a lógica de vizinhanças do GVNS diretamente na fase de geração de população inicial do BRKGA, criando indivíduos já "refinados" e combinando, busca populacional com metaheurística de solução única.

   Essas linhas de pesquisa devem ampliar a aplicabilidade do método e reforçar sua contribuição à pesquisa operacional na indústria siderúrgica.

Referências

Akiba, T., Sano, S., Yanase, T., Ohta, T., e Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160. URL https://doi.org/10.1287/ijoc.6.2.154.

Gonçalves, J. F. e Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487–525. URL https://link.springer.com/article/10.1007/s10732-010-9143-1.

Hansen, P., Mladenović, N., e Urošević, D. (2006). Variable neighborhood search and local branching. *Computers & Operations Research*, 33(10):3034–3045. URL https://doi.org/10.1016/j.cor.2005.02.023.

Keha, A. K., Özkan Kuş, e Taşkın, H. M. (2009). Mixed integer programming formulations for single-machine scheduling problems with sequence-dependent setup times. *European Journal of Operational Research*, 198(3):662–670.

Ma, C., Kong, M., Pei, J., e Pardalos, P. M. (2018). Brkga-vns for parallel-batching scheduling on a single machine with step-deteriorating jobs and release times. In Nicosia, G., Pardalos, P., Giuffrida, G., e Umeton, R., editors, *Machine Learning, Optimization, and Big Data*, p. 414–425, Cham. Springer International Publishing. ISBN 978-3-319-72926-8.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100. URL https://doi.org/10.1016/S0305-0548(97)00031-2.

Nogueira, T. H., Carvalho, C. R. V. d., Ravetti, M. G., e Souza, M. C. d. (2019). Analysis of mixed integer programming formulations for single machine scheduling problems with sequence dependent setup times and release dates. *Pesquisa Operacional*, 39:109–154.

Nogueira, T. H., Ramalhinho, H. L., de Carvalho, C. R., e Gomez Ravetti, M. (2022). A hybrid vns-lagrangean heuristic framework applied on single machine scheduling problem with sequence-dependent setup times, release dates and due dates. *Optimization Letters*, p. 1–20.

Rosa, B., Souza, M., e de Souza, S. (2018). Algorithms based on vns for solving the single machine scheduling problem with earliness and tardiness penalties. *Electronic Notes in Discrete Mathematics*, 66:47–54. ISSN 1571-0653. URL https://www.sciencedirect.com/science/article/pii/S1571065318300532. 5th International Conference on Variable Neighborhood Search.

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br

Sá, J. V. d. P., Tzitas, A. A., Sánchez, J. V. C., Paiva, D. N., e Souza, M. C. d. (2024). Vns no sequenciamento da etapa de têmpera em uma siderurgia. In *Anais do LVI Simpósio Brasileiro de Pesquisa Operacional*, Fortaleza. Galoá. Acesso em: 03 Maio 2025.

Tzitas, A. A., Sá, J. V. d. P., Sánchez, J. V. C., Otiai, F. S., e Souza, M. C. d. (2024). Otimização do sequenciamento da produção no processo de têmpera em uma siderurgia. In *Anais do LVI Simpósio Brasileiro de Pesquisa Operacional*, Fortaleza. Galoá. Acesso em: 03 Maio 2025.

Ye, M., Bartolini, E., e Schneider, M. (2024). A general variable neighborhood search for the traveling salesman problem with time windows under various objectives. *Discrete Applied Mathematics*, 346:95–114.

DOI: 10.59254/sbpo-2025-212131
https://proceedings.science/p/212131?lang=pt-br