LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

VNS NO SEQUENCIAMENTO DA ETAPA DE TÊMPERA EM UMA SIDERURGIA

João Victor de Paula Sá
Vallourec
Belo Horizonte, MG - Brasil
joao.sa@vallourec.com

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

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
daninascimento120@gmail.com

Maurício Cardoso de Souza
Universidade Federal de Minas Gerais
Belo Horizonte, MG - Brasil
mauriciodesouza@gmail.com

RESUMO

No processo produtivo das indústrias existem etapas que devem ser priorizadas no planejamento da produção. O presente trabalho tem como objetivo propor estratégias de resolução para o problema de sequenciamento na etapa de têmpera de uma indústria siderúrgica. Trata-se de um sequenciamento em máquina única com tempos de *setup* dependentes da sequência. A aplicação do algoritmo *Variable Neighborhood Search* (VNS), além do desenvolvimento de uma Heurística de Análise de Variância (HAV), resultou em melhorias significativas na minimização da data de término do cronograma em comparação com os métodos tradicionais utilizados pelo programador da produção. As estratégias demonstraram ser eficazes em todas as instâncias testadas, gerando soluções suficientemente boas em tempo reduzido. Os resultados destacam a eficiência e a utilidade prática das técnicas de Pesquisa Operacional no sequenciamento da produção.

PALAVRAS CHAVE. Otimização, Sequenciamento, Metaheurística, VNS, Siderurgia.

Tópicos: POI - PO na Indústria, MH – Metaheurísticas, OC - Otimização Combinatória, AD&GP – PO na Administração e Gestão da Produção

ABSTRACT

In the production process of industries, there are stages that must be prioritized in production planning. The present work aims to propose resolution strategies for the scheduling problem in the heat treatment stage of a steel industry. It is a single-machine problem with sequence-dependent

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

setup times. The application of the Variable Neighborhood Search (VNS) algorithm, along with the development of a Variance Analysis Heuristic (HAV), resulted in significant improvements in minimizing the schedule completion date compared to traditional methods used by the production analyst. The strategies proved to be effective in all tested instances, generating sufficiently good solutions in reduced time. The results highlight the efficiency and practical usefulness of Operational Research techniques in scheduling problems.
KEYWORDS. Optimization, Scheduling, Metaheuristics, VNS, Steel Industry.

Paper topics: POI – OR in Industry, MH – Metaheurístics, OC – Combinatorial Optimization, D&GP – OR in Administration and Production Management

1. Introdução

A busca por maior eficiência e competitividade é um desafio constante enfrentado por diversas indústrias. Com o avanço da Indústria 4.0, a tecnologia desempenha um papel fundamental na transformação dos processos produtivos, tornando-se essencial para aprimorar decisões e impulsionar a produtividade. Autores como Venâncio [2022] destacam a crescente integração das áreas de tecnologia e, diante deste cenário, a Pesquisa Operacional emerge como uma área capaz de potencializar reduções de custo, assim como promover uma maior aderência entre setores de planejamento e operação.

Este trabalho concentra-se nos desafios associados ao sequenciamento de ordens produtivas em um horizonte de curto prazo, no contexto de uma siderurgia. O foco está na etapa de têmpera, que pode ser definido como o aquecimento ou resfriamento de materiais com o objetivo de modificar suas propriedades mecânicas, sem afetar sua forma original. Atualmente, essa etapa é identificada como um ponto crítico na usina em questão, onde a capacidade produtiva do processo estabelece limitações à produção. Sendo assim, um eventual aumento na produtividade, em decorrência de uma melhor ordenação de um conjunto de operações (*jobs*), resulta em uma economia significativa.

Objetiva-se minimizar a soma dos tempos de preparação entre materiais subsequentes, caracterizando um problema com *setup* dependente da sequência, como visto em Bigras et al. [2008]. O *setup* é calculado em função da diferença de temperatura no processamento de dois materiais consecutivos, pois há um tempo para resfriamento ou aquecimento dos fornos em que são submetidos. Para um planejamento de aproximadamente uma semana, temos em média cerca de trinta a quarenta ordens. Atualmente, gasta-se um tempo significativo de um analista de planejamento para que uma sequência boa o suficiente seja encontrada. Absorver o dinamismo da operação, entregando cronogramas otimizados em um tempo satisfatório, é uma relevante necessidade do setor de planejamento da produção.

O desafio consiste em reduzir o *makespan*, isto é, o tempo total necessário para concluir um conjunto de ordens, levando em conta as restrições específicas do processo. Na abordagem proposta, todas as operações que ocorrem na têmpera são consideradas como uma máquina única, os *jobs* possuem um tempo de processamento e devem ser realizados de forma sequencial, onde a próxima tarefa só pode começar após a conclusão da atual. Considerando a disponibilidade de recursos, é crucial levar em conta as *release dates*, que indicam o momento em que uma tarefa está apta a ser executada.

Para garantir uma melhor aderência do planejamento otimizado aos processos operacionais, foram consideradas algumas premissas específicas. Dado que o planejamento da produção é realizado de forma contínua, é necessário fixar um primeiro *job* que represente o estado atual da máquina. Além disso, algumas ordens de produção, devido às suas características físicas, precisam passar pela têmpera duas vezes, sendo definidas como ordens de dupla têmpera. Estas ordens

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

são tratadas como dois *jobs* distintos, mas com os mesmos tempos de processamento e *setups* para outros *jobs*.

Além do *makespan*, os programadores consideram outros fatores essenciais, como os intervalos para manutenções preventivas. A ferramenta de otimização desenvolvida também oferece soluções alternativas que levam em conta pequenas variações neste objetivo. Isso permite maior flexibilidade, especialmente quando há necessidade de priorizar ordens atrasadas. Desta forma, garante-se que a tomada de decisão seja otimizada e simultaneamente adaptável, fornecendo uma robusta capacidade de resposta às dinâmicas do ambiente produtivo.

Para abordar o problema, o algoritmo de Busca em Vizinhança Variável (VNS - *Variable Neighborhood Search*), proposto por Mladenović e Hansen [1997], foi selecionado por apresentar bons resultados em condições semelhantes ao da problemática em questão. Após adaptar o VNS ao nosso contexto, foi proposta uma Heurística de Análise de Variância (HAV) com o objetivo de fornecer uma boa solução inicial. Além disso, após a adaptação de formulações matemáticas propostas por Nogueira et al. [2019] e Bigras et al. [2008], uma estratégia de resolução complementar foi utilizada, na qual a solução encontrada pelo VNS é fornecida como solução inicial para um modelo de programação linear inteira mista, visando obter um bom *primal bound* na busca pela solução ótima. Os responsáveis pelo sequenciamento do processo validaram os resultados e os aumentos de produtividade foram mensurados após a aplicação do trabalho desenvolvido, contribuindo para a eficiência do processo.

O artigo está estruturado da seguinte forma. Na Seção 2 é discutido o referencial teórico utilizado. A Seção 3 apresenta os principais métodos desenvolvidos nas construções das heurísticas. Resultados computacionais em instâncias reais são apresentados na Seção 4 e as conclusões do trabalho na Seção 5.

2. Referencial Teórico

O problema de sequenciamento em questão pode ser tratado como o *Single Machine with Setup Time Problem* (SMSP), caracterizado como um problema de otimização combinatória. Diversos estudos têm explorado diferentes abordagens metaheurísticas ou híbridas para lidar com as complexidades inerentes a ele e a algumas de suas variantes.

Nogueira et al. [2022] propôs uma heurística híbrida utilizando relaxação lagrangeana de forma integrada com o VNS para o SMSP, considerando datas de disponibilidade e datas de entrega dos *jobs*. Embora a função objetivo seja centrada na minimização do atraso, os resultados obtidos mostram que o VNS apresenta bom desempenho em problemas similares. No que tange a outras metaheurísticas aplicadas a esse problema, Bahalke et al. [2010] investiga o uso de algoritmos genéticos híbridos para resolvê-los. Embora não abordem diretamente janelas de tempo, suas técnicas de resolução, como utilizar a busca tabu como solução inicial para o algoritmo genético, podem ser aproveitadas no caso de outras metaheurísticas.

Dado que o problema em questão apresenta similaridades com variações do *Traveling Salesman Problem* (TSP), em especial sua variante *Traveling Salesman Problem with Time Windows* (TSPTW) que considera janelas de tempo, também foi realizada uma revisão em artigos que tratam esses problemas. López-Ibáñez et al. [2013] demonstra a viabilidade de adaptar métodos que apresentam bons resultados para o TSP em suas variantes, citando heurísticas como *Beam-ACO* e *Compresed Annealing* em seu trabalho.

Estudos como o de Mustu e Eren [2018] demonstraram a eficácia do VNS na resolução do SMSP. Ye et al. [2024] aplica o *General Variable Neighborhood Search* (GVNS) ao TSPTW, visando minimizar diferentes objetivos, incluindo custo, *makespan* e duração da rota. Os resultados mostram que o GVNS trouxe soluções de qualidade que competem com as heurísticas do estado da arte para variantes desse problema.

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

O VNS foi escolhido como o método a ser aplicado devido à sua eficácia e versatilidade apresentada em abordagens similares. Enquanto muitas metaheurísticas são valorizadas por gerarem soluções de qualidade em tempos computacionais aceitáveis, o VNS se distingue pelo seu método robusto de exploração do espaço de soluções através de mudanças na vizinhança, o que o torna eficaz para problemas como o SMSP e o TSPTW.

3. Aplicação do método VNS

O VNS opera em dois passos principais: o *shaking* e a busca local. O *shaking* é um mecanismo de perturbação que gera uma nova solução a partir da solução atual, movendo-se para uma vizinhança diferente. Esse movimento é aleatório e controlado por um parâmetro de vizinhança ($\mathcal{N}_l$), que define a intensidade ou a amplitude da perturbação. O objetivo é escapar de ótimos locais e explorar outras áreas do espaço de soluções que podem conter soluções melhores. Já o passo de busca local envolve explorar a vizinhança (N_k) de uma solução inicial X, aplicando melhorias incrementais e iterando esse processo até que não se possa mais encontrar uma solução melhor ou um critério de parada seja atingido. Aplica-se o procedimento de busca local em soluções pertubadas pelo *shaking*, garantindo assim a exploração eficiente do espaço de soluções na busca de ótimos locais e, potencialmente, globais.

A partir de uma solução inicial X_0 obtida através de uma heurística construtiva, que será explicada na Seção 3.3, e da amplitude máxima de perturbação da vizinhança l_{max}, inicia-se o algoritmo. A cada iteração, aplica-se a função de *shaking* para obtenção da solução perturbada X'. Esta solução serve como entrada no método de busca local, que retorna a solução X''. Sempre que uma melhoria no *makespan* é encontrada, a melhor solução atual X é substituída por X'' e a amplitude da vizinhaça do *shaking* ($\mathcal{N}_l$) é reiniciada. Se não houver melhoria, incrementa-se a amplitude. O algoritimo é interrompido quando não houver melhoria na amplitude máxima da vizinhança de *shaking* ($\mathcal{N}_{l_{Max}}$), ou quando o tempo limite de execução é atingido.

A Seção 3.1 aprofunda a explicação do método de *shaking*. Neste trabalho também utiliza-se uma estrutura de vizinhança variável na busca local, mais precisamente o *Variable Neigborhood Descent* (VND), como detalhado na Seção 3.2. O pseudocódigo do algoritmo desenvolvido é apresentado a seguir.

Algorithm 1: Variable Neighborhood Search (VNS)

Input: X_0: Solução inicial; l_{max}: Amplitude máxima de perturbação da vizinhança do *shaking*;
Output: X: Melhor solução encontrada

$X \leftarrow X_0$
$l \leftarrow 1$
while $l \neq L_{max}$ **do**
  $X' \leftarrow$ *Shaking*(X, l)
  X" $\leftarrow$ *Busca Local*(X')
  if *Makespan*$(X'') <$ *Makespan*(X) **then**
    $X \leftarrow X$"
    $l \leftarrow 1$
  end
  else
    $l \leftarrow l + 1$
  end
end
return X

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

3.1. *Shaking*

Para garantir uma diversificação adequada, o tamanho máximo do segmento selecionado para o *shaking* varia conforme o valor de l para a vizinhança $\mathcal{N}_l$. Neste trabalho, limitamos amplitudes de pertubação até l_{Max}. Esta restrição é projetada para equilibrar a necessidade de introduzir mudanças substanciais na solução e as pertubações feitas ainda permançam em uma vizinhança de soluções com um certo nível de confiança por estar próxima de uma solução já cerficada como boa, seja pela heurística construtiva ou pela busca local.

É importante destacar que o *shaking* é conduzido de forma a respeitar as restrições de viabilidade impostas pelo problema, incluindo as regras relacionadas às ordens de dupla têmpera. Sendo assim, o movimento não apenas promove a diversificação, mas também assegura que as soluções resultantes sejam viáveis e possam ser posteriormente melhoradas por meio de operações de busca local. Desta forma, o algoritmo proposto não irá retornar ou encontrar uma solução inviável sob nenhuma condição, garantindo que sua melhoria sempre permanecerá dentro do espaço viável de solução.

Suponhamos que uma solução é representada por uma sequência de *jobs*. Uma operação de *shaking* pode ser exemplificada da seguinte forma:

1. **Escolha da vizinhança**: Suponha que a vizinhança $\mathcal{N}_l$ seja definida a partir da reordenação de um segmento de tamanhho l de uma solução.
2. **Perturbação da solução atual**: Se a solução atual é dada pela sequência [J_1, J_2, J_3, J_4, J_5], e $l = 3$, seleciona-se um segmento de tamanho 3 e aplica-se uma função que reorganiza aleatoriamente este vetor, por exemplo:
   - Selecionar o segmento [J_2, J_3, J_4] e trocar as posições dos *jobs* de maneira aleatória, resultando em [J_1, J_4, J_3, J_2, J_5].
3. **Nova solução**: A nova solução [J_1, J_4, J_3, J_2, J_5] é então utilizada para a etapa de busca local.

3.2. Busca Local

Após a obtenção da solução perturbada X', realiza-se uma busca local baseada no método *Variable Neighborhood Descent* (VND). O pseudocódigo da implementação do método é apresentado a seguir.

Algorithm 2: Busca Local baseada no VND

Input: X_0: Solução inicial, k_{max}: tamanho máximo da vizinhança para a busca local
Output: X: Melhor solução encontrada

```
k ← 2
X ← X_0
while k ≤ k_max do
    X' ← k-swap(X, k)
    if Makespan(X') < Makespan(X) then
        X ← X'
        k ← 2
    end
    else
        k ← k + 1
    end
end
return X
```

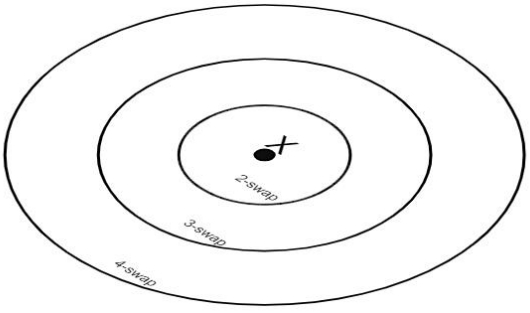
DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

A função *k-swap* determina como exploramos a vizinhança da solução X. Foram implementados três operadores para esta função:

- ***2-Swap***: Troca as posições de dois *jobs* na sequência, retornando a melhor solução dentre estas possibilidades.
- ***3-Swap***: Seleciona três *jobs* para trocar de posições, totalizando seis combinações possíveis. A função calcula o *makespan* para cada uma dessas combinações e retorna a sequência com o menor entre eles.
- ***4-Swap***: Similar ao *3-Swap*, mas considera quatro *jobs*. Há vinte e quatro combinações possíveis de posicionamento e o algoritmo seleciona a que resulta no menor *makespan*. Por ser uma busca mais custosa que a anterior, aplicamos uma lógica de *first neighbor*, ou seja, ao invés de calcularmos o *makespan* de todas as combinações e escolher a melhor, calcula-se proceduralmente e a primeira solução que apresentar melhoria em relação a solução inicial X é retornada.

A Figura 1 ilustra regiões do espaço de solução delimitados a partir de cada *k-swap*. Neste trabalho, adotamos um $k_{max} = 4$ em decorrência do número de combinações aumentar fatorialmente em relação à k.

Figura 1: Vizinhança de uma solução X



A Figura 2 ilustra a busca local por meio de um exemplo. A partir de uma solução X, foi encontrada uma solução melhor X' utilizando o operador *2-swap*. Em seguida, o *2-swap* é acionado novamente, mas sem sucesso na busca por uma melhoria e, portanto, aplica-se o *3-swap*. Desta vez obtém-se uma solução melhor, então o método é reiniciado a partir do operador *2-swap*. O ciclo é interrompido quando não há melhoria nos operadores *2, 3 e 4-swap* a partir de uma mesma solução. Esta forma de explorar a vizinhança se beneficia do tempo de cálculo dos seus operadores, valorizando melhorias em regiões de vizinhança menores, onde as buscas são mais rápidas.

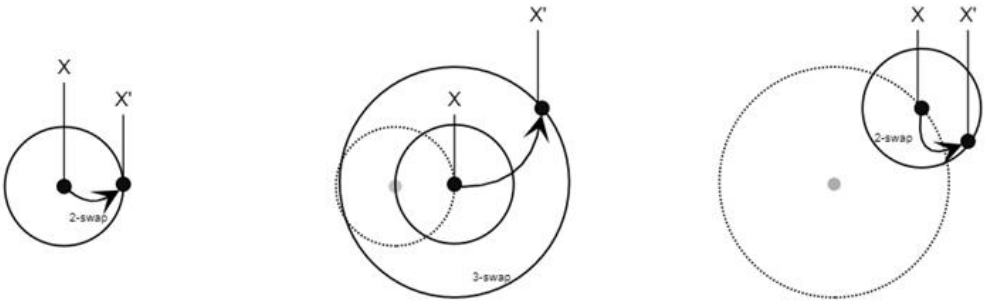
3.3. Heurística de Análise de Variância (HAV)

Dada a importância de uma solução inicial para o método, representada pelo X_0 no algoritmo 1, foi desenvolvida uma heurística construtiva baseada na análise de variância das posições dos *jobs* na sequencia. A HAV concentra-se na avaliação dos tempos de *setup* de cada *job* em relação aos demais. Esta avaliação identifica os *jobs* mais críticos, definidos por aqueles com maior impacto no *makespan*. Sendo assim, um *job* que possui menor criticidade é aquele que apresenta menor variância (c_j) no seu tempo de *setup* em relação aos outros.

O método inicia a partir da ordenação crescente dos *jobs* por c_j. Em seguida, uma sequência inicial aleatória é definida. Para aprimorar a solução, cada *job* é submetido a uma análise

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

Figura 2: Exemplo de um passo na Busca Local



de inserção de posição. Tenta-se realocar cada *job* dentro da sequência inicial, começando pela primeira posição. Se uma alteração resultar em redução do *makespan*, a inserção é realizada. Caso não haja melhoria, o processo avança para a posição seguinte, repetindo o processo até que todas as possibilidades tenham sido avaliadas. Se não houver nenhuma melhoria, o *job* mantém sua posição original. Este método é aplicado sequencialmente a todos os *jobs* na lista ordenada por c_j. Essa estratégia visa posicionar os *jobs* mais críticos em termos de *setup* por último. O pseudocódigo da HAV é apresentado a seguir.

Algorithm 3: Heurística de Análise de Variância (HAV)

Input: S: *jobs* ordenados por variância (c_j)
Output: X: Melhor solução encontrada

$X_0 \leftarrow$ Solução Aleatória
$X \leftarrow X_0$
for *job* $\in S$ **do**
  for $i \in$ *posições de* X_0 **do**
    $X' \leftarrow$ Sequência após mover *job* para nova posição i em X
    if *Makespan*$(X') <$ *Makespan*(X) **then**
      $X \leftarrow X'$
    end
  end
end
return X

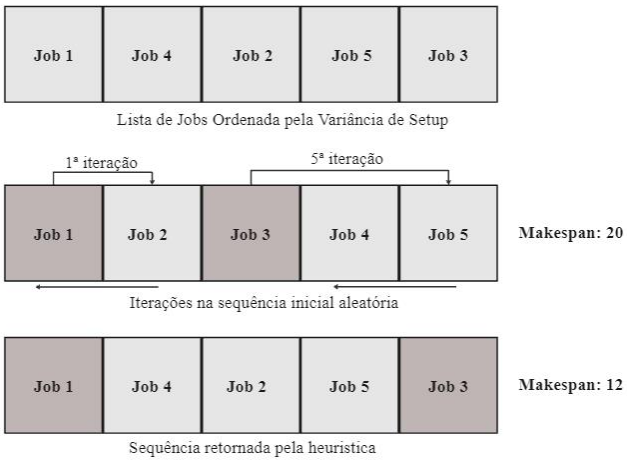
Como pode ser visto na Figura 3, o primeiro *job* a ser realocado vai ser aquele que tem a menor variância de *setup*, o *Job 1*. Já o último *job* a ser inserido vai ser o com maior variância, o *Job 3*. Nota-se que como a prioridade do *Job 1* é menor, ele pode voltar para a posição inicial dependendo das inserções dos outros *jobs* em outras posições. É importante destacar que a sequência resultante da heurística sempre será viável, respeitando todas as restrições do problema.

4. Resultados Computacionais

Os algoritmos propostos foram implementados na linguagem de programação *python* e ambiente de desenvolvimento *Spyder*. Para a realização dos testes, foram considerados oito cronogramas reais de produção, disponibilizados pelos responsáveis do sequenciamento da têmpera. Os horizontes de planejamento variam entre dois e quinze dias. Todas as execuções foram geradas em um recurso computacional de *16,0 GB* de memória *RAM* e processador *Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz*.

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

Figura 3: Esquema da inserção de *jobs* na HAV



Na primeira fase de testes, o objetivo foi comparar a performance do algoritmo VNS desenvolvido e do sequenciamento realizado pelo programador. Na Tabela 1 apresenta-se os resultados obtidos para cada uma das instâncias. Todas as instâncias foram executadas pelo algoritmo em um tempo limite máximo de dez minutos.

Tabela 1: Comparação dos *makespans* entre VNS e Programador

Instância	**N° de *jobs***	**Makespan [min]**		**Ganho [h]**
		VNS	**Programador**	
A_28j	28	7564.97	7862.45	4.96
B_50j	50	16591.07	16591.07	0.00
C_36j	36	13332.75	13420.20	1.46
D_16j	16	3031.51	3132.10	1.68
E_16j	16	3870.42	4142.19	4.53
F_33j	33	13406.77	13916.13	8.49
G_45j	45	15617.61	16194.02	9.61
H_26j	26	10252.54	10667.79	6.92

Nota-se que houveram ganhos significativos em aumento de capacidade produtiva em praticamente todas as instâncias. Apenas a B_50 apresentou o mesmo *makespan*.

O objetivo da otimização proposta é minimizar o *makespan*, promovendo maior eficiência produtiva e aumentando a aderência entre o planejamento e as operações da têmpera. Nesse sentido, o algoritmo desenvolvido gera várias sequências com mesmo valor de *makespan* e oferece ao programador múltiplas opções de ordenação dos *jobs*. Estas alternativas podem contribuir em decisões sob circunstâncias não planejadas, que envolvam disponibilidade de recursos, falhas em equipamentos ou especificidades das ordens processadas. A Figura 4 apresenta, para cada instância, o número de sequências viáveis fornecidas pela ferramenta de otimização, com o mesmo valor de *makespan*.

Embora o número de respostas distintas com o mesmo *makespan* possa variar entre as instâncias do problema, essa variedade oferece uma vantagem significativa para a aplicação prática na indústria. Essa diversidade de opções proporciona ao programador uma gama de alternativas

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

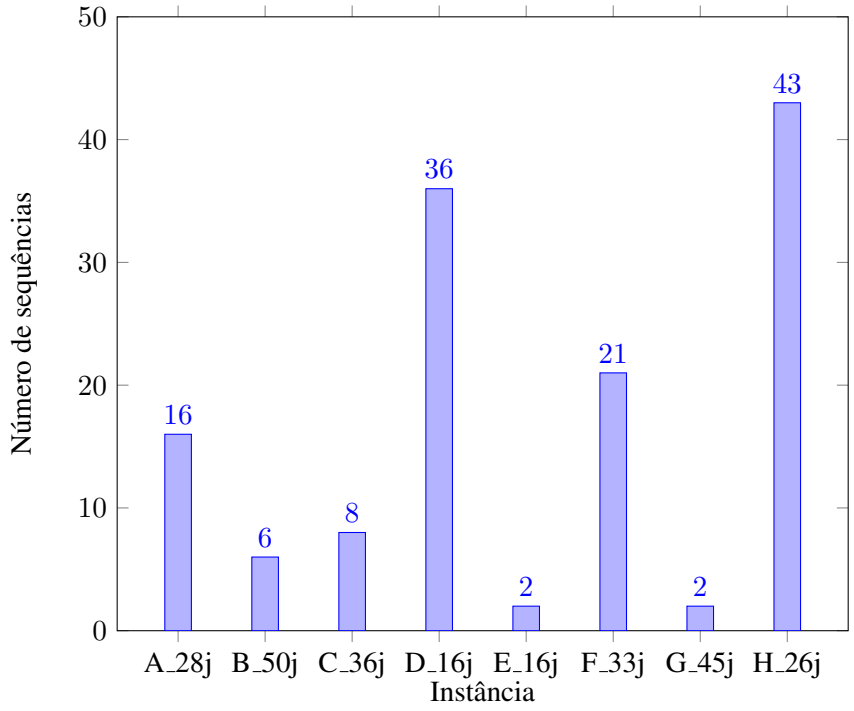


Figura 4: Número de sequências diferentes com mesmo *makespan*

igualmente eficazes, permitindo uma resposta mais ágil e adaptável às demandas e restrições do ambiente produtivo.

Investigou-se a resolução do problema apresentado utilizando programação linear inteira mista, através de duas formulações matemáticas propostas por Nogueira et al. [2019] e Bigras et al. [2008] e adaptadas para este problema por Tzitas et al. [2024], em um trabalho que está em desenvolvimento. Estas formulações foram denominadas FA1 e FA2, respectivamente. Utilizou-se os modelos desenvolvidos como base de comparação para o VNS, conforme apresentado na Tabela 2. O tempo limite de execução das formulações para os testes foi de uma hora, enquanto o do algoritmo VNS permaneceu fixado em dez minutos. Para o cálculo dos gaps obtidos a partir da solução do VNS, selecionou-se o melhor *dual bound* entre as formulações. Nota-se uma melhoria no *gap* em todas as instâncias nas quais os modelos não conseguiram provar a otimalidade.

Tabela 2: Comparação entre os gaps de otimalidade FA1, FA2 e VNS

Instância	**FA1**	**FA2**	**VNS**
A_50j	110%	0.24%	0.06%
B_50j	3.50%	47.27%	0.57%
C_36j	5.57%	4.91%	3.18%
D_16j	8.82%	0%	0%
E_16j	21.39%	0%	0%
F_33j	45.49%	2.76%	2.08%
G_45j	268.89%	2.41%	1.86%
H_26j	0%	53,2%	0%

Visando melhorar o desempenho dos modelos exatos, fornecemos a resposta do VNS

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

como solução inicial para as formulações FA1 e FA2. Esta estratégia foi utilizada na tentativa de conseguir menores *gaps* de otimalidade para o mesmo tempo de execução pré-determinado. As Tabelas 3 e 4 apresentam os resultados obtidos para cada uma das instâncias.

Tabela 3: Comparação entre VNS+FA1 e FA1

Instância	Tempo[s]		Makespan[min]		GAP	
	VNS+FA1	FA1	VNS+FA1	FA1	VNS+FA1	FA1
A_28j	3600	3600	7560.46	8263.07	95.67%	110.01%
B_50j	30	3600	16 591.07	17 067.79	0.00%	3.49%
C_36j	3600	3600	13 332.75	13 641.13	3.21%	5.57%
D_16j	3474	3600	3031.51	3031.51	0.00%	8.82%
E_16j	3600	3600	3870.42	3873.22	21.39%	32.27%
F_33j	3600	3600	13 409.92	13 977.38	40.00%	45.49%
G_45j	3600	3600	15 606.64	17 236.29	222.21%	268.89%
H_26j	163	70	10 252.54	10 252.54	0.00%	0.00%

Tabela 4: Comparação entre VNS+FA2 e FA2

Instância	Tempo[s]		Makespan[min]		GAP	
	VNS+FA2	FA2	VNS+FA2	FA2	VNS+FA2	FA2
A_28j	3505	3600	7560.46	7560.46	0.00%	0.24%
B_50j	3600	3600	16 591.07	21 442.43	13.00%	47.27%
C_36j	3600	3600	13 332.75	13 528.04	5.00%	6.91%
D_16j	3	5	3031.51	3031.51	0.00%	0.00%
E_16j	4	60	3870.42	3870.42	0.00%	0.00%
F_33j	3600	3600	13 409.92	13 496.32	2.35%	2.76%
G_45j	944	3600	15 606.64	15 702.65	0.00%	2.41%
H_26j	3600	3600	10 252.54	11 182.87	41.40%	53.20%

Os resultados revelam que a adoção do método complementar de resolução demonstrou eficácia em comparação com as formulações individuais. Para as instâncias que não alcançaram a solução ótima, observou-se uma redução significativa no *gap* de otimalidade. Além disso, para a maioria das instâncias o *makespan* foi reduzido.

Observa-se na Tabela 4 que foi possível provar a otimalidade das instâncias A_28j e G_45j antes de atingir o tempo limite de execução do modelo. Estes casos demonstram o impacto positivo desta abordagem complementar.

5. Conclusão

Neste trabalho, foram exploradas estratégias para otimizar o sequenciamento da etapa de têmpera em uma indústria siderúrgica. O foco principal foi a redução do *makespan*, visando aumentar capacidade e maximizar a eficiência deste processo aplicando técnicas de Pesquisa Operacional. O algoritmo do VNS foi adaptado ao problema de sequenciamento e uma heurística construtiva foi desenvolvida na busca de uma solução inicial. Além disso, uma abordagem complementar foi utilizada, no qual a solução obtida pelo VNS torna-se um limite superior para a resolução do problema via modelos de programação linear inteira mista.

Após a realização de testes utilizando instâncias reais, foi possível analisar ganhos significativos de produtividade nas soluções fornecidas pelo VNS em comparação com os métodos

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

anteriores. Além disso, múltiplas soluções distintas fornecidas pelo algoritmo com mesmo *makespan* auxiliam o programador da têmpera em circunstâncias não planejadas, contribuindo para a adaptabilidade ao ambiente dinâmico da produção.

Em uma segunda fase de testes, observou-se os ganhos obtidos a partir da aplicação da metaheurística desenvolvida como solução inicial dos modelos. Essa abordagem complementar apresentou um bom desempenho na busca por soluções de qualidade em um tempo máximo de resolução pré-determinado.

Os resultados da ferramenta de suporte à decisão foram validados pelos responsáveis do sequenciamento da têmpera, demonstrando uma significativa relevância prática na eficiência do planejamento da produção. Foi possível aumentar a capacidade operacional e se adaptar de forma mais ágil ao ritmo dinâmico das operações. Destaca-se que os tempos necessários para obter os sequenciamentos propostos pelo algoritmo foram consideravelmente menores em comparação aos métodos anteriores.

Como trabalhos futuros, o objetivo será sequenciar um cronograma de médio prazo, que abrange um horizonte temporal mais extenso. Nessa perspectiva, o aumento significativo do número de *jobs* pode levar à redução da qualidade do algoritmo VNS e tornar necessária a implementação de melhorias no método. Adicionalmente, será necessário incorporar as datas de conclusão das ordens a serem produzidas, o que pode ampliar a complexidade do problema e ressaltar a importância da busca por estratégias eficazes dentro do campo da Pesquisa Operacional.

Referências

Bahalke, U., Yolmeh, A. M., e Shahanaghi, K. (2010). Meta-heuristics to solve single-machine scheduling problem with sequence-dependent setup time and deteriorating jobs. *The International Journal of Advanced Manufacturing Technology*, 50:749–759.

Bigras, L.-P., Gamache, M., e Savard, G. (2008). The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Optimization*, 5(4):685–699.

López-Ibáñez, M., Blum, C., Ohlmann, J. W., e Thomas, B. W. (2013). The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization. *Applied Soft Computing*, 13(9):3806–3815.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.

Mustu, S. e Eren, T. (2018). The single machine scheduling problem with sequence-dependent setup times and a learning effect on processing times. *Applied Soft Computing*, 71:291–306.

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.

Tzitas, A. A., Sá, J. V. d. P., Otiai, J. V. C., Sánchez, Shoei, F., e Souza, M. C. d. (2024). Otimização do sequenciamento da produção no processo de tempera em uma siderurgia. Trabalho em Desevolvimento.

DOI: 10.59254/sbpo-2024-193488
https://proceedings.science/p/193488?lang=pt-br

Venâncio, C., V. e Machado (2022). Cartilha de convergência ot-it. Instituto Brasileiro de Petróleo e Gás. Comissão de Tecnologia e Inovação – Rio de Janeiro: IBP. Acessado: 25/03/2024.

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-2024-193488
https://proceedings.science/p/193488?lang=pt-br