Relatório Final de Pesquisa Operacional: Resolução do Problema de Blocking Flowshop

Gabriel Davi / Giordani Andre

Agosto de 2025

1 Links

1. Clique aqui para acessar o Codigo
2. Clique aqui para acessar o Slide

2 Introdução ao artigo

O artigo aborda o Problema de Sequenciamento em Flowshop com Bloqueio (do inglês, Blocking Flowshop Scheduling Problem - BFSP), um problema de otimização combinatória NP-completo de grande relevância em sistemas de produção onde não existem pulmões (buffers) intermediários entre as máquinas. Neste cenário, uma tarefa, ao finalizar seu processamento em uma máquina, a bloqueia até que a máquina subsequente esteja livre. O objetivo do trabalho é minimizar o tempo de fluxo total (do inglês, Total Flowtime), que corresponde à soma dos tempos de completação de todas as tarefas na última máquina.

3 Revisão da Literatura

O artigo de referência, "Minimização do Tempo Total de Fluxo no Problema de Blocking Flowshop com Uso de GRASP Reativo" de Gelli et al., apresenta uma abordagem de meta-heurística para resolver o problema BFSP com o objetivo de minimizar o tempo de fluxo total. Sendo um problema de classificação NP-difícil, o uso de heurísticas é justificado para encontrar boas soluções em tempo computacional viável.

A abordagem proposta pelos autores é um GRASP (Greedy Randomized Adaptive Search Procedure) Reativo. A principal característica desta variação do GRASP é a sua capacidade de autoajuste do parâmetro α, que controla o nível de aleatoriedade na construção da solução, adaptando-se ao longo da execução com base na qualidade das soluções geradas.

O procedimento é composto por duas fases principais:

- **Fase Construtiva:** Utiliza uma versão aleatorizada da heurística FF, proposta originalmente por Fernandez-Viagas e Framinan.
- **Busca Local:** Após a construção, uma solução é refinada através de buscas locais do tipo *Interchange* (troca de duas tarefas) e *Insertion* (realocação de uma tarefa).

Os autores validaram seu método utilizando as instâncias de benchmark de Taillard e compararam seus resultados com os do estado da arte, obtidos por Tasgetiren et al., demonstrando a alta performance de sua abordagem.

4 Desenvolvimento da Solução

Para a resolução do problema, foi desenvolvido um conjunto de algoritmos em Python, utilizando as bibliotecas 'numpy' para análise estatística e 'tabulate' para a exibição de resultados. A estratégia adotada seguiu uma abordagem em duas fases: uma fase construtiva para gerar uma solução inicial de qualidade, seguida por uma fase de refinamento utilizando duas técnicas de meta-heurística distintas.

4.1 Heurística Construtiva: NEH

Como ponto de partida, foi implementada a clássica e eficiente heurística de Nawaz, Enscore e Ham (NEH). O funcionamento da função `heuristica_neh` consiste em:

1. Calcular o tempo total de processamento para cada tarefa em todas as máquinas.
2. Ordenar as tarefas em ordem decrescente com base nessa soma.
3. Inserir iterativamente cada tarefa da lista ordenada na sequência parcial, testando todas as posições possíveis e escolhendo aquela que resulta no menor tempo de fluxo total (calculado pela função `calcular_total_flowtime_BLOCKING`).

Esta heurística foi escolhida por sua simplicidade de implementação e sua conhecida eficácia em gerar boas soluções iniciais para problemas de flowshop.

4.2 Fase de Refinamento: Meta-heurísticas

A solução gerada pela NEH foi então utilizada como entrada para duas abordagens de refinamento distintas, permitindo uma comparação de eficácia.

4.2.1 Busca Local com Interchange

A função `busca_local_interchange` implementa uma busca local iterativa. Partindo da solução NEH, ela explora sistematicamente a vizinhança de troca (swap) de todas as tarefas. A cada iteração, ela avalia todas as trocas possíveis

e aplica a que gera a maior melhoria (estratégia *Best Improvement*). O processo se repete até que nenhuma troca possa melhorar a solução atual, atingindo assim um ótimo local.

4.2.2 Simulated Annealing (SA)

A função `simulated_annealing` implementa a meta-heurística de Recozimento Simulado. Assim como a busca local, ela utiliza a vizinhança de *Interchange* para gerar novas soluções. No entanto, sua principal característica é a capacidade de aceitar soluções piores com uma certa probabilidade, que diminui conforme a "temperatura" do sistema baixa. Este mecanismo permite que o algoritmo escape de ótimos locais e explore uma área mais ampla do espaço de soluções. Os parâmetros de resfriamento (temperatura inicial, final e fator alfa) foram definidos com valores padrão da literatura para garantir um número consistente de iterações.

5 Resultados Encontrados

Para avaliar o desempenho dos algoritmos implementados, os resultados foram comparados com os valores de BKS (Best Known Solutions) da literatura através do Desvio Percentual Relativo (RPD), cuja fórmula é dada por:

$$RPD = \frac{(CustoObtido - CustoBKS)}{CustoBKS} \times 100\%$$

Os testes foram executados com um tempo limite de 480 segundos (8 minutos) por instância, o mesmo utilizado no artigo de referência, para garantir uma comparação justa. A seguir, apresentamos um resumo dos resultados para o grupo de instâncias 50x20.

Table 1: Resumo dos resultados para o grupo 50x20

Métrica	**NEH + Interchange**	**NEH + SA**
ARPD (%)	52.51	53.92
Tempo Médio (s)	1.35	0.07
Iterações (Média)	14	1919

5.1 Análise Comparativa

A partir dos resultados, observamos que:

- **NEH + Interchange:** Este método se mostrou extremamente rápido, encontrando um ótimo local em poucos segundos e com um número muito baixo de iterações. Ele consistentemente melhora a solução da NEH, mas seu RPD final ainda é muito alto (superior a 50%), indicando que ele fica preso em ótimos locais de baixa qualidade.

- **NEH + Simulated Annealing:** Este método apresentou um comportamento misto. Em algumas instâncias, foi capaz de superar o resultado do Interchange, mas em muitas outras não conseguiu melhorar a solução inicial da NEH. O número de iterações foi fixo em 1919, determinado pelo cronograma de resfriamento, e não pelo tempo limite.

- **Comparação com o Artigo:** Ambos os métodos implementados tiveram um desempenho significativamente inferior ao do GRASP Reativo do artigo (que obteve ARPD próximo de 0%). A principal razão para essa diferença é a estratégia do algoritmo: nosso método é *single-start*, refinando uma única solução inicial. Em contraste, o GRASP é *multi-start*, gerando e refinando milhares de soluções diferentes dentro do tempo limite de 8 minutos, o que aumenta drasticamente a chance de encontrar soluções de elite.

6 Conclusão

O objetivo deste trabalho foi implementar e avaliar o desempenho de heurísticas para a resolução do Problema de Sequenciamento em Flowshop com Bloqueio (BFSP) com o objetivo de minimizar o tempo total de fluxo. Foram implementadas com sucesso a heurística construtiva NEH e duas estratégias de refinamento: uma Busca Local baseada em Interchange e a meta-heurística Simulated Annealing.

Os resultados obtidos, quando comparados com o estado da arte, demonstraram que, embora os algoritmos funcionem corretamente e forneçam melhorias sobre uma solução inicial, eles não são suficientes para atingir a qualidade das soluções encontradas por meta-heurísticas mais complexas e com maior tempo de execução, como o GRASP Reativo apresentado no artigo de referência. O estudo confirma a complexidade do problema e evidencia o clássico trade-off entre tempo de execução e qualidade da solução na otimização de problemas NP-difíceis.