Case modelo V_0

Felipe Sinnecker

Fevereiro 2024

1 Problema

O problema proposto é uma versão simplificada de um problema clássico, trata-se do job-shop problem (JSP). De maneira geral temos um conjunto de tarefas $J_1, \cdots, J_n$ a serem realizadas e um conjunto de máquinas $M_1, \cdot, M_m$ para realiza-las. Cada tarefa precisa ser alocada em uma máquina, que irá levar um determinado tempo para concluir a tarefa. O objetivo é organizar a ordem em que as tarefas serão alocadas a cada máquina, baseado em algum critério como custo ou tempo, de forma que o critério escolhido seja otimizado.

Para o modelo V_0 apresentado, estamos querendo saber em que ordem e maquina os tecidos serão alocados, com o objetivo de minimizar o tempo total para a etapa de enfesto ser concluída. Desconsiderando a existência das mesas no modelo, temos 2 máquinas de enfesto disponíveis E_1, E_2 e 4 ordens de produção diferentes $OP1, OP2, OP3, OP4$. Os Dados foram utilizados para a modelagem foram os tempos de setup de cada máquina assim como o tempo médio de enfesto para cada ordem:

Máquina	Tempo de setup
E1	2
E2	3

Ordem de Produção	Tempo médio de enfesto
OP1	12
OP2	10
OP3	8
OP4	4

Baseado nos vídeos disponíveis assim como as informações fornecidas no enunciado, podemos assumir que a etapa de enfesto não pode ser dividida em pequenas etapas e realizada em mais de uma máquina, portanto uma vez que o enfesto de uma ordem começa devemos terminar antes de realizar outra. Então o modelo deve considerar os tempos médios de enfesto assim como os tempos de setup para obter uma sequência de ordens para cada uma das máquinas realizar a etapa de enfesto, sendo que todas as ordens de produção devem ser concluídas, não podendo ser interrompidas uma vez que comecem. Podemos utilizar o modelo clássico para o JSP, utilizando variáveis binárias para indicar a ordem, com variáveis inteiras para obter os valores de tempo em que cada ordem termina.

2 Modelo

O modelo clássico de job-shop utiliza variáveis binárias para indicar a ordem em que as tarefas são executadas em cada máquina, sendo que existe uma tarefa extra 0, podemos considerar as tarefas a serem realizadas como de 1 a 4, esta tarefa extra representa o início da ordem de cada máquina, assim como o retorno a ela representa o encerramento das tarefas na máquina.

Dados e conjuntos

Temos os seguintes conjuntos para o problema:

$$\begin{aligned} J_0, &\quad \text{ordem de produção com a ordem extra} \\ J, &\quad \text{ordem de produção} \\ M, &\quad \text{máquinas} \end{aligned}$$

O conjunto J_0 representa tanto o conjunto J com as ordens de produção de 1 a 4, como a ordem extra 0. O conjunto M representa as máquinas 1 e 2 de enfesto. Para os dados fornecidos do problema temos:

$$\begin{aligned} st_m, &\quad m \in M \qquad \text{tempo de setup de cada máquina } m \\ TE_i, &\quad i \in J \qquad \text{tempo médio de enfesto de cara ordem } i \end{aligned}$$

Variáveis

Podemos modelar o problema utilizando variáveis binárias:

$$\begin{aligned} x_{i,j,m} \in \{0,1\} \quad i,j \in J_0 \quad m \in M \\ y_{i,m} \in \{0,1\} \quad i \in J \quad m \in M \end{aligned}$$

A variável $x_{i,j,m}$ indica que a etapa de enfesto da ordem i procede a da ordem j na máquina m. A variável $y_{i,m}$ indica que a ordem i está sendo processada na máquina m. Temos também as variáveis inteiras:

$$\begin{aligned} E_i \in Z \quad i \in J_0 \\ T \in Z \end{aligned}$$

As variáveis E_i representam o tempo final da etapa de enfesto de cada ordem i, enquanto a variável T serve para modelar o maior tempo $\max_i\{E_i\}$.

Função Objetivo

A função objetivo é dado pelo valor do maior tempo, logo é o valor da variável Z:

$$\min Z$$

Restrições

Para o conjunto de restrições temos:

$$E_0 = 0 \tag{1}$$

$$x_{i,i,m} = 0, \quad \forall i \in J_0 \tag{2}$$

$$\sum_{m \in M} y_{i,m} = 1, \quad \forall i \in J \tag{3}$$

$$\sum_{j \in J_0} x_{i,j,m} = y_{i,m}, \quad \forall i \in J \quad \forall m \in M \tag{4}$$

$$\sum_{i \in J_0} x_{i,j,m} = y_{j,m}, \quad \forall j \in J \quad \forall m \in M \tag{5}$$

$$\sum_{j \in J} x_{0,j,m} \leq 1, \quad \forall m \in M \tag{6}$$

$$E_j \geq E_i + st_m + TE_j - (1 - x_{i,j,m})\theta, \quad \forall i \in J_0 \quad \forall j \in J \quad \forall m \in M \tag{7}$$

$$T \geq E_i, \quad \forall iinJ \tag{8}$$

$$E_i \geq 0, \quad \forall i \in J \tag{9}$$

A restrição (1) garantem que o tempo de enfesto da variável auxiliar seja 0, sendo ela a primeira da ordem sempre. A restrição (2) garante que a não possamos partir de uma ordem para ela mesma. A restrição (3) garante que cada ordem seja executada em somente uma das máquina. As restrições (4) e (5) garantem que cada ordem tenha apenas um sucessor e um predecessor na máquina em que está sendo executada. A restrição (6) garante que a ordem extra de cada máquina tenha apenas um sucessor, dando início a sequência de tarefas em cada máquina. A restrição (7) utiliza um valor θ grande para garantir que caso a ordem j seja a sucessora da ordem i na máquina m o enfesto é realizado após o anterior somado aos tempos de setup e de enfesto correspondentes ($x_{i,j,m} = 1 \rightarrow E_j \geq E_i + st_m + TP_j$), quando $x_{i,j,m} = 0$ temos que $\theta >> E_i + st_m + TP_j$ e logo a restrição (9) vale. A restrição (8) representa o valor máximo dos E_i e restringe a variável da função objetivo ao $\max_i E_i$.

3 Implementação Gurobi

O modelo foi implementado com a linguagem de programação julia, utilizando o pacote JuMP para escrever o modelo. O solver escolhido para resolver foi o Gurobi. Os dados do problema foram inseridos de forma manual.

Modelo

O JuMP permite que todas as restrições sejam escritas de forma igual a representada na seção de restrições, após o modelo ser compilado o Gurobi resolve

Resultado

Temos que o resultado dado pelo Gurobi foi a sequência de ordens [OP3,OP2] na máquina $E1$ e [OP1,OP4] na máquina $E2$. Podemos ver facilmente que essa é solução ótima pois ambos os tempos de tarefa são iguais, ou seja, diminuir o tempo de uma das mesas implica em aumentar o tempo da outra necessariamente. Podemos ver também que a ordem não importa em cada máquina individualmente, uma vez que mudar a ordem não altera o tempo final.

4 Implementação Metaheurística

Para resolver o modelo v_0 podemos utilizar a técnica de simulated annealing. O primeiro passo para aplicação de técnicas de MarkovChain-MonteCArlo é a definição de uma cadeia de Markov. Devido a experiências anteriores com um problema de roteamento de veículos, as soluções são representadas como vetores. Neste caso cada solução s é um vetor de vetores, onde cada vetor representa uma máquina e contem a ordem de tarefas a ser executada:

$$s = [V_1, V_2]$$
$$V_i = [v_1^i, \cdots, v_m^i], \quad v_j^i \in \mathcal{N}$$

Tome o exemplo fornecido,s = [[1,4],[2,3]], a máquina E_1 realiza o enfesto das ordens 1 e 4 nesta ordem e a máquina E_2 realiza o enfesto das ordens 2 e 3 nesta ordem.

Vizinhança

Definido o espaço de estados, temos que definir uma estrutura de vizinhança. Podemos transformar o estado s em um vetor único, colocando todas as ordens juntas e simbolizando com um 0 a separação das máquinas:

$$[[2,3,7,1,4][5,8,6]] \rightarrow [2,3,7,1,4,0,5,8,6]$$

Para o novo vetor podemos aplicar uma série de transformações, cada uma gerando um vizinho diferente:

Troca 1

Podemos escolher uma ordem de maneira uniforme, escolhendo uma nova posição para ela também de maneira uniforme. Exemplo mudando a ordem 3:

$$[2,3,7,1,4,0,5,8,6] \rightarrow [2,7,1,4,0,3,5,8,6] \rightarrow [[2,7,1,4][3,5,8,6]]$$

Troca 2

Podemos escolher 2 ordens de maneira uniforme, invertendo suas posições de lugar. Exemplo mudando as ordens 4 e 8:

$$[2,3,7,1,4,0,5,8,6] \rightarrow [2,3,7,1,8,5,0,4,6] \rightarrow [[2,7,1,8,5][4,6]]$$

Importance Sampling

A fim de selecionar vizinhos de melhor qualidade, ainda mantendo a possibilidade de se amostrar um vizinho que seja diversificado do atual, foi implementada uma busca local gulosa. A nova busca selecionava ordem de maneira uniforme e modificava ela de lugar, calculando os valores de função objetivo para cada um dos possíveis vizinhos, por fim retornava o melhor vizinho. Para aumentar a probabilidade de um vizinho de alta qualidade ser selecionado definimos um parâmetro $p \in [0, 1]$, onde a cada passo do SA com probabilidade p amostramos um vizinho de qualidade dado pela busca gulosa, com probabilidade $1 - p$ usamos uma das 2 trocas aleatórias escolhida de maneira uniforme para gerar um vizinho.

Simulated Annealing

A técnica de Simulated Annealing (SA) consiste em construir uma cadeia de Markov utilizando Metropolis Hastings e a distribuição de Blotzman para transicionar pelos estados. O método é baseado no processo térmico de recozimento, no qual um sólido tem a temperatura aumentada e resfriado gradativamente para que o material passa para um estado de menor energia interna. Assim para um problema de otimização, a energia interna é análoga a função objetivo, então transicionamos de um estado a outro buscando diminuir gradativamente o valor da função objetivo em busca de um ótimo, ainda permitindo que estados que sejam piores que a solução atual sejam visitados baseados na temperatura.

Resultado

Temos que o resultado dado pelo SA tem o mesmo valor ótimo do Gurobi, mesmo com uma solução diferente. Este modelo é simples e por isso mesmo limitando a apenas 100 iterações do SA o método converge.