Case modelo V_1

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_1 apresentado, estamos querendo saber em que ordem os tecidos serão alocados na máquina de enfesto, assim como a ordem dos cortes. O objetivo é minimizar o tempo total para a etapa de corte ser concluída. Desconsiderando a existência das mesas no modelo, temos 1 máquina de enfestoE_1, 1 máquina de corte C_1 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,o tempo médio de enfesto e de corte para cada ordem:

Máquina	Tempo de setup
E1	2
C1	4

Ordem de Produção	Tempo médio de enfesto	Tempo médio de corte
OP1	18	4
OP2	14	4
OP3	12	6
OP4	6	6

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. A etapa de corte só pode começar caso sua respectiva etapa de enfesto já tenha sido concluída, estamos considerando que a ordem de corte não precisa ser igual a ordem de enfesto. Todas as ordens de produção devem ser concluídas. Podemos utilizar o mesmo modelo anterior $v0$ adaptando as variáveis.

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_1) e 2 (C_1). Para os dados fornecidos do problema temos:

st_m,	$m \in M$	tempo de setup de cada máquina m
TE_i,	$i \in J$	tempo médio de enfesto de cara ordem i
TC_i,	$i \in J$	tempo médio de corte de cara ordem i

Variáveis

Podemos modelar o problema utilizando variáveis binárias:

$$x_{i,j,m} \in \{0,1\} \quad i,j \in J_0 \quad m \in M$$

A variável $x_{i,j,m}$ indica que a etapa de enfesto da ordem i procede a da ordem j na máquina $m = 1$, o mesmo vale para a etapa de corte da máquina $m = 2$. Temos também as variáveis inteiras:

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

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

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:

$$E_0 = 0 \tag{1}$$
$$C_0 = 0 \tag{2}$$
$$x_{i,i,m} = 0, \quad \forall i \in J_0 \tag{3}$$
$$\sum_{j \in J_0} x_{i,j,m} = 1, \quad \forall i \in J \quad \forall m \in M \tag{4}$$
$$\sum_{i \in J_0} x_{i,j,m} = 1, \quad \forall j \in J \quad \forall m \in M \tag{5}$$
$$\sum_{j \in J} x_{0,j,m} \leq 1, \quad \forall m \in M \tag{6}$$
$$E_j \geq E_i + st_1 + TE_j - (1 - x_{i,j,1})\theta, \quad \forall i \in J_0 \quad \forall j \in J \tag{7}$$
$$C_j \geq C_i + st_2 + TC_j - (1 - x_{i,j,2})\theta, \quad \forall i \in J_0 \quad \forall j \in J \tag{8}$$
$$C_i \geq E_i + st_2 + TC_i, \quad \forall i \in J \tag{9}$$
$$T \geq C_i, \quad \forall i \in J \tag{10}$$
$$E_i, C_i \geq 0, \quad \forall i \in J \tag{11}$$

As restrições (1) e (2) garantem que os tempos de enfesto e corte da variável auxiliar sejam 0, sendo ela a primeira da ordem sempre. A restrição (3) garante que a não possamos partir de uma ordem para ela mesma. 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 restrições (7) e (8) utilizam um valor θ grande para garantir que caso a ordem j seja a sucessora da ordem i na máquina m a tarefa é realizado após o anterior somado aos tempos de setup e de da tarefa correspondentes, quando $x_{i,j,m} = 0$ temos que a restrição (11) vale. A restrição (9) impede que a etapa de corte comece antes da etapa de enfesto chegar ao fim. A restrição (10) representa o valor máximo dos C_i e restringe a variável da função objetivo ao $\max_i C_i$.

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.

Resultado

Temos que o resultado dado pelo Gurobi foi a sequência de enfestos [OP4,OP3,OP1,OP2] na máquina $E1$, com os tempos de enfesto finais [8,22,42,58] respectivamente.

A sequência de cortes [OP3,OP4,OP1,OP2] na máquina $C1$ possui tempos finais [32,42,50,66] respectivamente, com 66 sendo o valor ótimo do problema. Podemos ver facilmente que essa é solução ótima pois, o tempo total da etapa de enfesto é sempre o mesmo uma vez que uma ordem sempre começa depois da outra o tempo dop fim da última ordem sempre será 58. Basta então que a última ordem a ser concuída seja a de menor tempo de corte, podendo ser tanto OP1 quanto OP2, neste caso somamos o tempo de corte ao valor 58 que junto ao tempo de setup leva 8 minutos totalizando 66.

4 Implementação Metaheurística

Para resolver o modelo v_0 podemos utilizar a mesma técnica de simulated annealing para resolver, fazendo uma pequena mudança na estrutura de vizinhança. As soluções continuam 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. Remos apenas uma máquina de enfesto e 1 máquina de corte, então temos 2 vetores, ambos com uma sequência contendo todas as ordens:

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

Tome o exemplo fornecido,s = [[1,4,2,3],[1,4,2,3]], ambas as máquinas E_1 e C_1 realizam as ordens na mesma sequência.

Vizinhança

Podemos escolher 1 dos vetores, seja o da maquina de enfesto ou de corte e aplicar as seguintes operações

Troca 1

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

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

Troca 2

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

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

Importance Sampling

A fim de selecionar vizinhos de melhor qualidade, ainda mantendo a possibilidade de se amostrar um vizinho que seja diversificado do atual, utilizamos a mesma técnica do modelo v_0. A nova busca seleciona um dos vetores, de corte ou enfesto para alterar de maneira uniforme. Escolhe uma ordem para alterar, 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.

Resultado

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