Open Shop and Job Shop
SAAD Practical Assignment – Group #1 ID A

Danilo Brandão, Gabriel Carvalhal e Wagner Ceulin

Mestrado em Engenharia e Ciência de Dados – Universidade do Porto

Resumo. Neste trabalho abordou-se o problema de programação de tarefas nos ambientes FLOW SHOP SEQUENCING com 20 jobs e 5 máquinas e JOB SHOP SCHEDULING com 15 jobs e 15 máquinas conforme publicado por E. Taillard em "BENCHMARKS FOR BASIC SCHEDULING PROBLEMS". Foram analisadas as performances de duas estratégias de busca da solução ótima para o modelo CP (ou programação com restrições) em comparação com modelos MIP (programação linear inteira mista) para verificar a correção e eficiência.

Palavras-chave: flow shop, permutação, heurística, algoritmo de timing.

1 Definição dos Problemas e Contextualização

1.1 Introdução

Scheduling é a alocação de recursos compartilhados ao longo do tempo para atividades concorrentes.

Job Shop Scheduling (JSS) ou Job Shop Problem (JSP) é um problema de otimização popular em ciência da computação e pesquisa operacional. Isso se concentra na atribuição de tarefas a recursos em momentos específicos. Um Job Shop é um local de trabalho no qual existem várias estações de trabalho de uso geral e são usadas para executar uma variedade de trabalhos. A versão mais básica do JSSP é: os n jobs $J_1, J_2, \ldots, J_n$ de tempos de processamento variados precisam ser escalonados em m máquinas com poder de processamento variável, enquanto se tenta minimizar o *makespan*.

O *makespan* é a duração total do agendamento (ou seja, quando todos os trabalhos tiverem concluído o processamento). Cada trabalho consiste em uma sequência de tarefas que devem ser executadas em uma determinada ordem, e cada tarefa deve ser processada em uma máquina específica.

Restrições para o Problema de Job Shop: **(1)** nenhuma tarefa de um trabalho pode ser iniciada até que a tarefa anterior desse trabalho seja concluída; **(2)** uma máquina só pode trabalhar em uma tarefa por vez e **(3)** uma tarefa, uma vez iniciada, deve ser executada até a conclusão.

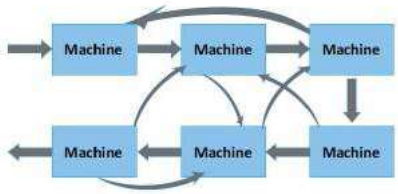

Fig. 1 - Ilustração do Job Shop Scheduling

Representação de JSS

O Gráfico de Gantt é uma maneira conveniente de representar visualmente uma solução do JSSP.

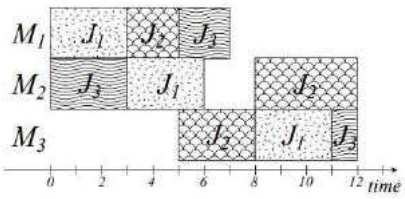

Fig. 2 - Representação de um problema 3x3.

Onde J_1-J_3 significam os Jobs e M_1-M_3 significam as Máquinas. O comprimento desta solução é 12, que é a primeira vez que todos os três trabalhos são concluídos.

Flow Shop Sequencing

O Flow Shop Sequencing é um caso especial de JSS, onde há uma ordem estrita de todas as operações a serem executadas em todos os jobs. A versão mais básica do FSS é: os *n* jobs J_1, J_2, …, J_n de tempos de processamento especificados variados precisam ser escalonados em *m* máquinas. A *i*-ésima operação do trabalho deve ser executada na *i*-ésima máquina. Nenhuma máquina pode realizar mais de uma operação simultaneamente.

As operações dentro de um trabalho devem ser executadas na ordem especificada. A primeira operação é executada na primeira máquina, depois (quando a primeira operação é concluída) a segunda operação na segunda máquina e assim por diante até a *n*-ésima operação. O objetivo é determinar o arranjo ideal, aquele com o menor tempo de execução total de trabalho possível.

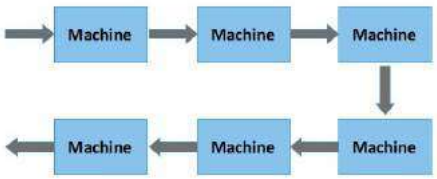

Fig. 3 - Ilustração do Flow Shop Scheduling

Job Shop vs. Flow Shop

A diferença fundamental entre como a sequência de tarefas é processada entre os dois agendamentos pode ser ilustrada abaixo.

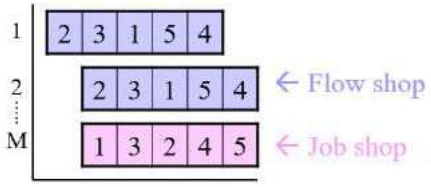

Fig. 4

Para resolver o problema de escalonamento, uma ampla gama de soluções tem sido proposta tanto em ciência da computação quanto em pesquisa operacional.

2 Propostas de Solução

2.1 Comparação entre MIP e Constraint Programming

Os métodos exatos procuram obter a solução ótima para o problema a partir da construção de modelos matemáticos (e.g. algoritmo de branch-and-bound) de otimização e da implementação de algoritmos específicos para a sua resolução. Contudo, devido à complexidade combinatória de alguns problemas, o tempo computacional necessário para a sua resolução torna-se muito elevado, sendo, geralmente, um tempo não-polinomial.

Já os métodos heurísticos são capazes de encontrar soluções viáveis em tempo de execução polinomial, mas não garantem a qualidade da solução encontrada. Estes métodos tentam adotar uma estratégia que equilibre a qualidade da solução obtida com o tempo total de processamento. Assim, o algoritmo, a cada iteração, aproxima-se da solução ótima. A qualidade da solução alcançada está diretamente relacionada ao tempo de execução.

A seguir apresentamos as soluções que desenvolvemos.

2.2 Soluções por Constraint Programming (Programação por Restrições)

Job Shop Scheduling (JSS): utilizamos o CP_MODEL da biblioteca ORTOOLS. **Ver código ref. 1.1 no ANEXO I**. Os resultados obtidos foram:

Optimal Schedule Length: 1231.0
Statistics (- conflicts: 12329, - branches: 15187, - wall time: 10,368533 s)
Parâmetros de entrada: nº das máquinas e tempos dos jobs

Flow Shop Sequencing: utilizamos o CP_MODEL da biblioteca DOCPLEX. **Ver código ref. 2.2 no ANEXO I**. Os resultados obtidos foram:

Optimal Schedule Length: 1278.0
Statistics (- branches: 6864012, - wall time: 3,77 s)
Parâmetros de entrada: nº de máquinas, nº de jobs e tempos dos jobs

2.3 Soluções por MIP

Job Shop Scheduling (JSS): utilizamos as bibliotecas ITERTOOLS e MIP. **Ver código ref. 1.2, ANEXO I e print ANEXO II.** Os resultados parciais obtidos foram:

Optimal Schedule Length: Entre 1500 e 1021.
Statistics (- nodes 4.217.936 - wall time: interrompido após 81442,28 s)
Parâmetros de entrada: nº da máquina, tempo do job, horizonte e seed.

p_{ik} – tempo de processamento da tarefa i na máquina k.
$i(1), \ldots, i(m)$ – roteiro de processamento da tarefa i nas m máquinas, ou seja, a sequência de máquinas em que as operações dessa tarefa são processadas.
M – número suficientemente grande.
C_{ik} – instante de término de processamento da tarefa i na máquina k.

Onde,

$$x_{ijk} = \begin{cases} 1 \text{ se a tarefa } i \text{ precede a tarefa } j \text{ na máquina } k \\ 0 \qquad\qquad\qquad\qquad\quad \text{caso contrário} \end{cases}$$

Sujeito as seguintes restrições:

$$C_{i,i(1)} \geq p_{i,i(1)}, \quad i = 1, \ldots, n$$

$$C_{i,i(k+1)} \geq C_{i(k)} + p_{i,i(k+1)}, \quad i = 1, \ldots, n, \quad k = 1, \ldots, m-1$$

$$C_{jk} \geq C_{ik} + p_{jk} - M(1 - x_{ijk}), \quad i = 1, \ldots, n, \quad j = 1, \ldots, n \;\; k = 1, \ldots, m$$

$$C_{ik} \geq C_{jk} + p_{ik} - Mx_{ijk}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n \;\; k = 1, \ldots, m$$

$$C_{max} \geq C_{i,i(m)}$$

$$T_{max} \geq C_{i,i(m)}$$

$$C_{ik} \leq T_{i,i(m)} + d_{i,i(m)}$$

$$T_{i,i(m)} \leq M * U_{i,i(m)}$$

Fig.5 - Algoritmo utilizado nesta resolução.

Flow Shop Sequencing: utilizamos a biblioteca GUROBIPY e o modelo proposto por Wilson [5]. **Ver código ref. 2.2 no ANEXO I.** Os resultados obtidos foram:

Optimal Schedule Length: 1278.0
Statistics (- branches : 3406, - wall time: 3,78 s)
Parâmetros de entrada: tempos dos jobs, em tabela com formato 5 x 20.

The model is a flow shop scheduling problem, presented in Wilson (1989), as following:

$$z = \min(s_{m,n} + \sum_{j=1}^{n} p_{m,j} z_{j,n})$$

subject to

$$\sum_{j=1}^{n} z_{j,i} = 1, \quad 1 \leqslant i \leqslant n$$

$$\sum_{i=1}^{n} z_{j,i} = 1, \quad 1 \leqslant j \leqslant n$$

$$s_{1,1} = 0$$

$$s_{1,i} + \sum_{j=1}^{n} p_{1,j} z_{j,i} - s_{1,i+1}, \quad 1 \leqslant i \leqslant n-1$$

$$s_{r,1} + \sum_{j=1}^{n} p_{r,j} z_{j,1} = s_{r+1,1}, \quad 1 \leqslant r \leqslant m-1$$

$$s_{r,i} + \sum_{j=1}^{n} p_{r,j} z_{j,i} \leqslant s_{r+1,i}, \quad 1 \leqslant r \leqslant m-1, \quad 2 \leqslant i \leqslant n$$

$$s_{r,i} + \sum_{j=1}^{n} p_{r,j} z_{j,i} \leqslant s_{r,i+1}, \quad 2 \leqslant r \leqslant m, \quad 1 \leqslant i \leqslant n-1$$

$$z_{j,i} \in \{0,1\}, \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant i \leqslant n$$

$$s_{r,i} \geqslant 0, \quad 1 \leqslant r \leqslant m, \quad 1 \leqslant i \leqslant n$$

Fig.6 - Algoritmo utilizado nesta resolução.

3 Comparação de resultados e desempenho

De forma a simplificar a comparação de desempenho para obtenção da solução ótima, apresentamos o quadro ilustrativo abaixo, com o resumo dos resultados encontrados nas diferentes simulações indicadas nos itens 2.2 e 2.3.

	JS CP	JS MIP	FS CP	FS MIP
Wall Time	8.32 s	>81442.28 s	3,77 s	3,78 s
Optimal Schedule Length	1231	Entre 1500 e 1021	1278	1278

As imagens abaixo ilustram a solução ótima encontrada por meio de Constraint Programming para os casos de Job Shop Scheduling e Flow Shop Sequencing.

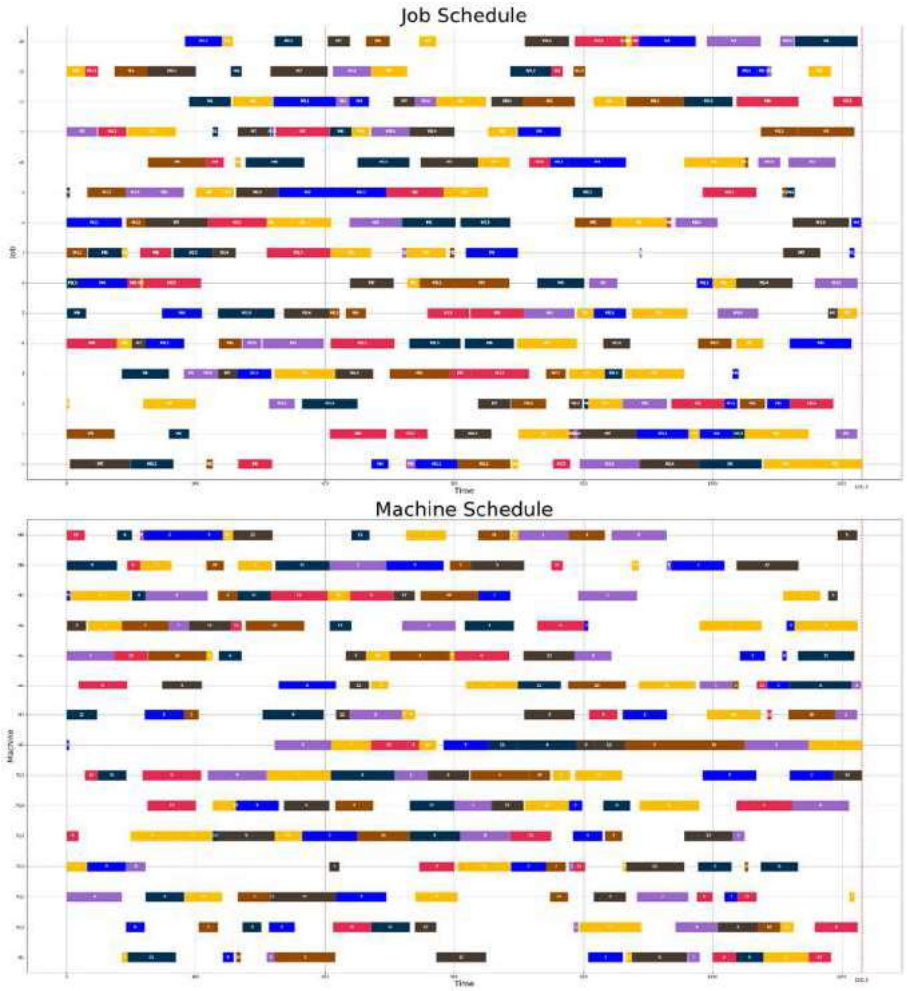

Fig.7 - Job Shop Scheduling - CP Programing Results

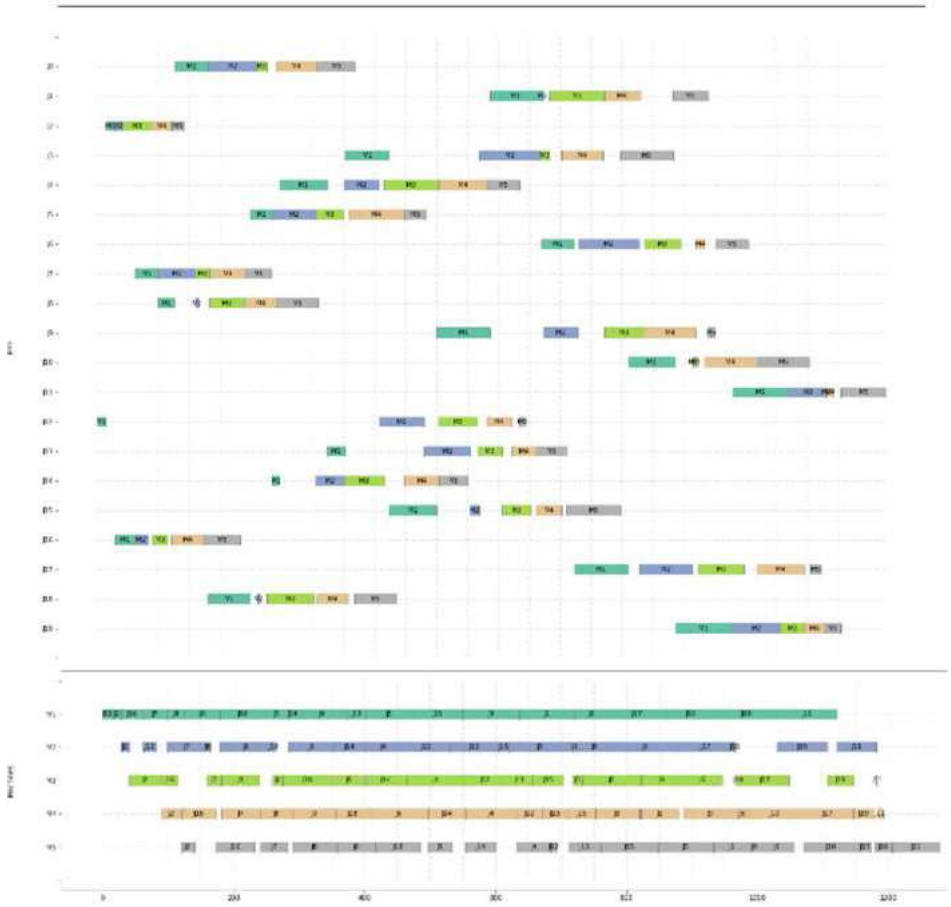

Fig.8 - Flow Shop Sequecing - CP Programing Results

4 Conclusão

Em cada teste realizado, depois de salvas as informações, foram comparados os resultados do melhor tempo obtido pela heurística em relação ao ótimo obtido pelo método exato. Constata-se que a relação tempo de processamento vs. resultados obtidos é mais vantajosa com a utilização do modelo Constraint Programming no script. , eEntretanto, essta vantagem se mostra mais evidente nos modelos maiores, devido ao aumento exponencial das iterações necessárias para se obter a solução exata (como pode ser visto no caso do JSS de 15 máquinas x 15 jobs).

Não foi possível encontrar a solução do modelo exato do Job Shop 15 máquinas x 15 jobs devido ao processamento envolvido [6]. O algoritmo se mostrou excelente no teste com modelagem 3 máquinas x 3 jobs, 4 máquinas x 4 jobs e 5 máquinas x 5 jobs (ver

figura abaixo e Notebook ANEXO III), mas no caso prático aqui estudado (15 máquinas e 15 jobs), interrompemos a execução após 22h e 30 minutos de processamento devido a restrições físicas.

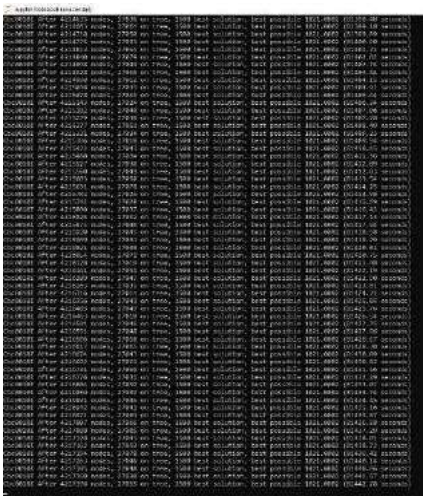

Fig.9 - Log do Jupyter na interrupção do processo.

Todos os resultados apresentados foram obtidos em um desktop com a seguinte configurações: • Sistema Operacional Microsoft Windows 10 Pro – Versão 10.0.19043 Compilação 19043 • Tipo do sistema PC baseado em X64 • Processador Intel(R) Core(TM) i7-4771 CPU @ 3.50GHz, 3501 Mhz, 4 Núcleos, 8 Processadores • Versão/data do BIOS American Megatrends Inc. 3503, 18/04/2018 • BaseBoard GRYPHON Z97 ARMOR EDITION (ASUS) • Memória Física (RAM) Instalada 32.0 GB • Placa Gráfica NVIDIA GeForce GTX 1650 4Gb.

Referências Bibliográficas e Websites

1. E. Taillard, "Benchmarks for basic scheduling problems", EJOR 64(2):278-285, 1993.
2. Chapter 7: Job-shop scheduling (pp. 134–160) Genetic algorithms in engineering systems Edited by A.M.S. Zalzala and P.J. Fleming IEE control engineering series 55 c 1997: The Institution of Electrical Engineers, ISBN: 0852969023
3. Job Shop Scheduling Problem (JSSP): An Overview by Chathurangi Shyalika, Nov 9, 2019 (https://medium.datadriveninvestor.com/job-shop-scheduling-problem-jssp-an-overview-cd99970a02f8)
4. https://python-mip.readthedocs.io/en/latest/bibliography.html
5. John M. Wilson, "Alternative Formulations of a Flow-shop Scheduling Problem", April 1989.
6. Nuno Manuel Boleo Teles de Jesus, "Programação da produção: Otimização de Layouts Industriais", Dissertação de Mestrado - Mestrado em Engenharia e Gestão Industrial do Instituto Politécnico do Porto, 2017.