Shop Scheduling

Applications

- **Model Factory-like Settings**
- **Also models packet routing**
- **...**

Basic Model: **Multiple machines. A jobs consist of operations, each operations has a**

- **processing time** p_{ij}
- **Machine on which to run** M_{ij}

Variants of Shop Scheduling

Basic Types

- **Job shop.** Each job consist of operations in a linear order
- **Flow shop.** Job shop, but the linear order is the same for each job. (assembly line)
- **Open shop.** Each job consists of unordered operations.

Other Constraints

- time between operations
  - mininimum time (e.g. cooling)
  - maximum time (e.g. hot potatoe)
- setup at machines (e.g. paint color)
- limited storage between machines

$F||C_{\max}$

First Question: Is it optimal to have each job go through the machines in the same order? (permutation schedule)

2 machines. Permutation schedule is optimal.

Example

j	p_{1j}	p_{2j}
1	3	6
2	10	1
3	3	2
4	2	4
5	8	8

What is the right algorithm?

SPT(I)- LPT(II)

Example

j	p_{1j}	p_{2j}
1	3	6
2	10	1
3	3	2
4	2	4
5	8	8

Algorithm:

- **Partition into two sets:**
  - **Set I has** $p_{1j} \leq p_{2j}$ **(1,4,5)**
  - **Set II has** $p_{1j} > p_{2j}$ **(3,2)**
- **Run Set I in SPT order by** p_{1j}
- **Run Set II in LPT order by** p_{2j}

For this problem: **4,1,5,2,3,**

Can use interchange arguments to show that this is optimal

- **Set I before Set II**
- **Set I in SPT order**
- **Set II in LPT order.**

More general flow shop

- **3 machines. There is an optimal permutations schedule.**
- **4 machines. Optimal schedule may not be a permutation schedule.**

$F|\text{perm}|C_{\max}$ as a mixed integer program

Decision variables: $x_{jk} = 1$ if job j is k th in sequence

Extra Variables:

- I_{ik} : idle time on machine i between jobs in positions k and $k+1$.
- W_{ik} : waiting time of job in position k between machines i and $i+1$.

Ideas

- Makespan is sum of
  - Processing time of first job on all machines
  - processing time of all jobs on machine m
  - Idle time on machine m
- Matching constraints to ensure that each job is in one position and each position has one job
- Relationship between idle time and waiting time constraints.
- Way to map variables so you can talk about k th job to run, rather than job indexed by j .

MIP

Processing time of k th job to run on machine i :

$$p_{i(k)} = \sum_{j=1}^{n} x_{jk} p_{ij}$$

Objective

$$\sum_{i=1}^{m-1} p_{i(1)} + \sum_{j=2}^{n} p_{mj} + \sum_{j=1}^{n-1} I_m j$$

Matching Constraints

$$\sum_{j=1}^{n} x_{jk} = 1 \quad k = 1 \ldots n$$

$$\sum_{k=1}^{n} x_{jk} = 1 \quad j = 1 \ldots n$$

Constraints relating idle and waiting time

$$I_{ik} + p_{i(k+1)} + W_{i,k+1} = W_{ik} + p_{i+1(k)} + I_{i+1,k} \quad \forall k, i$$

$$W_{i1} = 0 \forall i, \quad I_{1k} = 0 \forall k$$

.

Other Facts

- $F3||C_{\max}$ **is NP-complete.**
- $F3|\textbf{perm}|C_{\max}$ **is NP-complete.**
- **Easy case: all operations are the same size. Then flowshop with many objectives is easy.**

Slope Heuristic

Motivation: **Think about SPT(I)-LPT(II).**

- **Early jobs should be small on** M_1 **and large on** M_2 **.**
- **Late jobs should be large on** M_1 **and small on** M_2 **.**
- **Generalize to "slope". Larger slope should go earlier.**
- **Slope** $A_j = -\sum_{i=1}^{m}(m-(2i-1))p_{ij}$

Example

	J_1	J_2	J_3	J_4	J_5
M_1	**5**	**5**	**3**	**6**	**3**
M_2	**4**	**4**	**2**	**4**	**4**
M_3	**4**	**4**	**3**	**4**	**1**
M_4	**3**	**6**	**3**	**2**	**5**

Example

Example

	J_1	J_2	J_3	J_4	J_5
M_1	**5**	**5**	**3**	**6**	**3**
M_2	**4**	**4**	**2**	**4**	**4**
M_3	**4**	**4**	**3**	**4**	**1**
M_4	**3**	**6**	**3**	**2**	**5**

Example:Compute Slopes

	J_1	J_2	J_3	J_4	J_5
M_1	**5**	**5**	**3**	**6**	**3**
M_2	**4**	**4**	**2**	**4**	**4**
M_3	**4**	**4**	**3**	**4**	**1**
M_4	**3**	**6**	**3**	**2**	**5**
A_j	**-6**	**3**	**1**	**-12**	**3**