Simple Dispatch Rules

- We will first look at some simple dispatch rules: algorithms for which the decision about which job to run next is made based on the jobs and the time (but not on the history of jobs running, or by computing tentative schedules).
- These are also called greedy algorithms.

Goals:

- To recognize when simple dispatch rules apply.
- To prove that they are the correct algorithm.
- To analyze the running time of the algorithm.

$1||\Sigma C_j$

Example:

j	p_j
1	**5**
2	**3**
3	**10**
4	**8**
5	**4**

Questions:

- **What is the right algorithm?**
- **What is its running time?**
- **How do we prove it?**

$1||\sum C_j$

Example:

j	p_j
1	**5**
2	**3**
3	**10**
4	**8**
5	**4**

Questions:

- **What is the right algorithm? – SPT**
- **What is its running time?**
- **How do we prove it?**

$1||\Sigma C_j$

Example:

j	p_j
1	**5**
2	**3**
3	**10**
4	**8**
5	**4**

Questions:

- **What is the right algorithm?– SPT**
- **What is its running time? –** $O(n \log n)$
- **How do we prove it?**

$1||\Sigma C_j$

Example:

j	p_j
1	**5**
2	**3**
3	**10**
4	**8**
5	**4**

Questions:

- **What is the right algorithm?– SPT**
- **What is its running time? –** $O(n \log n)$
- **How do we prove it? – Interchange Argument**

Basic format of an interchange argument

- **Specify the simple dispatch rule X.**
- **Assume, for the purposed of contradiction, that you have an optimal schedule that does not obey the rule X.**
- **Find two specific jobs j and k that violate rule X.**
- **Show that if you interchange jobs j and k, then**
  - **The resulting schedule is still feasible.**
  - **The objective function value does not increase (for a minimization problem).**
- **You can then conclude that, via repeated swaps, there must exist an optimal schedule that satisfied rule X.**

Comment: **Even though the proof is boilerplate, you must provide enough mathematical detail that shows that your proof applies to the particular problem and the particular rule!**

$1||\Sigma w_j C_j$

Example:

j	p_j	w_j
1	**1**	**1**
2	**3**	**2**
3	**7**	**1**
4	**10**	**20**

Questions:

- **How can we figure out a simple dispatch rule?**
- **How do we prove it is correct – Interchange Argument**

$1||\Sigma w_j C_j$

Example:

j	p_j	w_j
1	**1**	**1**
2	**3**	**2**
3	**7**	**1**
4	**10**	**20**

Questions:

- **How can we figure out a simple dispatch rule?**
- **How do we prove it is correct – Interchange Argument**

Two answers to first question;

- **Experiment with small examples and develop a plausible rule.**
- **Start an exchange argument and see what you need to make it work.**

WSPT, Smith's rule

- **Want to have large weight, small processing time jobs early**
- **Schedule jobs in decreasing order of** w_j/p_j **.**

$1||\Sigma_j w_j(1-e^{-rC_j})$

- r **is a constant.**
- **For intuition:**

C_j	$(1-e^{-rC_j}$	$r=.1$
0	**0**	**0**
1	$1-e^{-r}$	**.10**
2	$1-e^{-2r}$	**.18**
3	$1-e^{-3r}$	**.25**
10	$1-e^{-10r}$	**.63**
100	$1-e^{-100r}$	**.99**

The optimal rule is to order by

$$\frac{w_j j e^{-rp_j}}{1-e^{-rp_j}}$$

Problems with chain precedence constraints

j	p_j	w_j
1	**3**	**6**
2	**6**	**18**
3	**6**	**12**
4	**5**	**8**
5	**4**	**8**
6	**8**	**17**
7	**10**	**18**

Chains: $1 \to 2 \to 3 \to 4$, $5 \to 6 \to 7$

- **Version 1: Once you start a chain, the whole chain must be completed.**
- **Version 2: Chains represent normal precedence constraints.**

What are the right algorithms?

Version 1: Whole Chain

j	p_j	w_j
1	**3**	**6**
2	**6**	**18**
3	**6**	**12**
4	**5**	**8**
5	**4**	**8**
6	**8**	**17**
7	**10**	**18**

Chains: $1 \to 2 \to 3 \to 4$, $5 \to 6 \to 7$

Choose the chain with the maximum

$$\frac{\Sigma\, w_j}{\Sigma\, p_j}$$

Version 2: Normal Chain Precendence Constraints

j	p_j	w_j	chain ratio	second round	third round
1	**3**	**6**	**2**	**x**	**x**
2	**6**	**18**	**8/3**	**x**	**x**
3	**6**	**12**	**12/5**	**2**	**2**
4	**5**	**8**	**11/5**	**20/11**	**20/11**
5	**4**	**8**	**2**	**2**	**x**
6	**8**	**17**	**25/12**	**25/12**	**x**
7	**10**	**18**	**43/22**	**42/22**	**9/5**

Chains: $1 \to 2 \to 3 \to 4$, $5 \to 6 \to 7$

Choose the chain prefix with the maximum

$$\frac{\Sigma\, w_j}{\Sigma\, p_j}$$

Adding release dates to a completion time problem.

$$1|r_j|\sum C_j$$

Example 1

j	r_j	p_j
1	**0**	**100**
2	**40**	**1**

Example 2

j	r_j	p_j
1	**0**	**100**
2	**40**	**1**
3	**101**	**1**

Problems with Deadlines

Deadlines

- **Can be hard (deadlines) or soft (due dates).**
- **Model Real Time scheduling.**

Example 1

j	p_j	d_j
1	**2**	**25**
2	**4**	**13**
3	**6**	**6**
4	**7**	**19**
5	**10**	**29**

Example 2

j	p_j	d_j
1	**1**	**5**
2	**3**	**6**
3	**6**	**7**
4	**4**	**15**

EDD - Earliest Due date

a.k.a. EDF - Earliest deadline first.

Theorem **If, for an instance, on one machine with processing times and deadlines, there is a schedule meeting all deadlines, then EDF meets all deadlines.**

What if you can't meet all deadlines:

- **One metric: lateness** $L_j = C_j - d_j$
- $1||L_{\max}$
- **Interpretation: if** $L_{\max}$ **is** 4**, then there is a schedule in which no job misses its deadline by more than** 4 **time units.**

Question: **Does EDD minimize** $L_{\max}$ **?**

Answer:

What if you can't meet all deadlines:

- **One metric: lateness** $L_j = C_j - d_j$
- $1||L_{\max}$
- **Interpretation: if** $L_{\max}$ **is** 4**, then there is a schedule in which no job misses its deadline by more than** 4 **time units.**

Question: **Does EDD minimize** $L_{\max}$ **?**

Answer: **YES**

- **Reason 1. Think about** $L_{\max} = t$ **as extending all deadlines by** t **.**
- **Reason 2. Interchange argument.**

Adding Release Dates. $1|r_j|L_{\max}$

Question: **Does EDD still produce an optimal schedule?**

j	r_j	p_j	d_j
1	**4**	**1**	**5**
2	**0**	**4**	**6**

Maybe some other dispatch rule still works?

Example 1

j	r_j	p_j	d_j
1	**0**	**5**	**11**
2	**4**	**1**	**6**
3	**5**	**1**	**6**

Example 2

j	r_j	p_j	d_j
1	**0**	**5**	**11**
2	**4**	**1**	**6**
3'	**6**	**1**	**7**