Scheduling: vocabulary

a: arrival time
C: computation time
d: deadline
s: starting time
f: finishing time
L = f - d: lateness
E = max(0, L): tardiness
X = d - a - C: slack

Scheduling: types

- Preemptive vs. non-preemptive.
- Synchronous (all processes arrive at once) or asynchronous (processes arrive one by one).
- Static (a, C, d of all processes known in advance) or on-line.
- Independent, or with dependency relation.
- Aperiodic vs. periodic vs. hybrid (mixed)

Scheduling Anomalies

- Theorem [Graham, 1976]:
  If a task set is optimally scheduled on a multiprocessor with some priority assignment, fixed execution times, and precedence constraints, then the following can increase the schedule length:
  - increasing the number of processors
  - reducing execution times
  - weakening the precedence relation

Scheduling anomalies - example

Adding one processor

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Decreasing the running time by 1

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>
Scheduling anomalies - example
Weakening the precedence relation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where's the catch?

- Theorem [Graham, 1976]:
  If a task set is optimally scheduled on a multiprocessor with some priority assignment, fixed execution times, and precedence constraints, then the following can increase the schedule length:
- Priorities are a bad idea. They confuse two concepts:
  - How important is something?
  - Should it be scheduled next?

Scheduling without Precedence Constraints

Earliest Due Date (EDD)

1|synch|Lmax

Use when:
- Synch: all jobs have r=0 (arrive immediately)
- Optimizes maximum lateness
- No precedence relations
- No preemption necessary

Earliest Due Date (EDD)

Given a set of processes 
\{[C_0, d_0], [C_1, d_1], \ldots, [C_n, d_n]\},
in order to minimize \(L_{\text{max}}\) it suffices to schedule the processes in order of increasing deadlines.

- EDD Algorithm:
  1) Sort the deadlines in increasing order
  2) Schedule the processes in order of increasing deadline.
- Schedulability criterion:
  Once ordered so that \(d_1 < d_2 < \ldots < d_n\), check that \(\sum_{k=1}^{m} C_k < d_k\) for all \(k \in \{1, \ldots, n\}\).

EDD Optimality

Recall: \(L = f - d\)
Assume \(d_a < d_b\)

<table>
<thead>
<tr>
<th>S</th>
<th>J_a</th>
<th>J_b</th>
<th>f_a</th>
<th>f_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>S'</td>
<td>J_a</td>
<td>J_b</td>
<td>f'_a</td>
<td>f'_b</td>
</tr>
</tbody>
</table>

What about the max latency in \(S'\)?

Case 1: if \(L'_a > L'_b\), note that \(D_a = T_a - d_a < f_a - d_a = L_a\).
Case 2: if \(L'_b > L'_b\), note that \(D_b = T_b - d_b < f_b - d_b < f_a - d_a = L_a\).
Earliest Deadline First (EDF) 
1|asynch|L_{\text{max}}

Use when:
- Tasks can have arbitrary arrival time.
- They are preemptable. This is important!

EDF Algorithm

[Horn]: to minimize L_{\text{max}}, always execute the process that has the earliest deadline

Proof of EDF Optimality

Scheduling with Precedence Constraints

Scheduling with precedence constraints (1 | prec, syn | L_{\text{max}}):
LDF (Latest Deadline First)

- Build the schedule from last to first, by recursively:
  - Let Q be the subset of jobs all whose descendants have been scheduled.
  - Pick the job p from Q that has the latest deadline.
  - Add p to the front of the schedule.
- Should be really called LDL (Latest deadline last).
- Optimal wrt. L_{\text{max}}.
LDF: Proof of Optimality

\[ S \triangleq A \mid J_a \mid B \mid J_b \]

\[ S' \triangleq A \mid B \mid J_a \mid J_b \]

\( J_b \) should be last according to LDF

\( L_{\text{max}} = \max \{ L_a, L_b, L_c, L_d \} \)

\( L'_a \) unchanged

\( L'_b \leq L_b \) as \( B \) starts earlier

\( L'_c \leq L_c \) as \( J_b \) starts earlier

\( L'_c = c - d_c \leq d_b - d_a \) (as \( d_a > d_b \))

So \( L_{\text{max}} \leq L_{\text{max}} \)

The argument is concluded by induction.

EDF with precedence constraints (1 | async, prec, preem | \( L_{\text{max}} \))

- Algorithm by Chetto, Silly, Bouchenouf (1990).
- Idea: transform timing constraints to enforce precedence (cannot start before predecessors, cannot pre-empt successors), then use EDF.

EDF with precedence constraints (1 | async, prec, preem | \( L_{\text{max}} \))

- When \( J_a < J_b \), then:
  - \( s_a \geq r_b \) (\( J_b \) must start after being released)
  - \( s_a \geq r_a + C_a \) (so \( J_a \) has time to finish)
- Thus, replace \( r_b \) by \( r^*_b = \max(r_b, r_a + C_a) \)

EDF with precedence constraints (1 | async, prec, preem | \( L_{\text{max}} \))

- Algorithm for release times:
  - For all roots \( J_a \) of dependency graph, let \( r^*_a = r_a \).
  - Select a task \( J_b \) such that all predecessors of \( J_b \) have already been modified; let \( H \) be this set.
  - Let \( r^*_b = \max \{ r_b, \max_{J \in H} (r^*_c + C_c) \} \)
- And recur.
**EDF: Modification of the Deadlines**

- Given Ja and Jb s.t. J_a < J_b, then the following conditions must hold:
  - \( f_a \leq d_a \) (or else we miss the deadline)
  - \( f_a \leq d_b - C_b \) (or there won't be enough time for \( a \) to finish).
- Hence, replace \( d_a \) by \( d_a^* = \min(d_a, d_b - C_b) \)

**Algorithm for deadline modification:**
- For all sinks \( J_a \) of dependency graph, let \( d_a^* = d_a \).
- Select a task \( J_b \) such that all successors of \( J_b \) have already been modified; let \( H \) be this set.
- Let \( d_a^* = \min\{d_a, \min_{c \in H} (d_c^* - C_c)\} \)
- And recur.

**EDF Modification Example**

![Diagram of EDF Modification Example 1](image1)

![Diagram of EDF Modification Example 2](image2)

![Diagram of EDF Modification Example 3](image3)
EDF Modification Example

EDF Modification Example

EDF Modification Example

EDF Modification Example

EDF Modification Example

Sometimes we can see right away that the problem is not schedulable
Another Example

EDF + prec: Proof of optimality

- The algorithm provides a translation EDF+Prec 1 modified-EDF.
- We can show that the translation preserves schedulability.
- First direction: if EDF+Prec is schedulable, then modified-EDF is also schedulable (the same schedule works).

EDF + prec: Proof of optimality

- Converse: if modified-EDF is schedulable, the same schedule shows that EDF+Prec is schedulable.
- Clearly, the schedule produced by modified EDF meets all release times and deadlines of the original problem; we must only prove that the precedence relation is respected.
- Assume \( J_2 \prec J_3 \) and \( C_2 > C_3 \). Then, \( D^*_2 < D^*_3 \), so EDF schedules \( J_2 \) before \( J_3 \).

Non-Preemptive Scheduling

- Jobs have release times, arrive dynamically, and there are no precedence relations.
- There are two kinds of algorithms:
  - Non-idle, that always execute a job as long as there is one ready
  - May-Idle, that may choose to do nothing even though there are jobs ready.
- There are cases when may-idle algorithms outperform non-idle ones....

Being idle is sometimes a good idea

EDF is best of non-idle 1|async|\( L_{\text{max}} \) algorithms

- This is a theorem by Jeffay, Stanat, and Martel (1991).
- Intuition: if you can't help being active, at least do the most useful thing at hand, that is, the one with the earliest deadline.
A periodic scheduling: summary

<table>
<thead>
<tr>
<th>sync</th>
<th>preempt</th>
<th>non-preempt</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDD</td>
<td>EDF</td>
<td>if non-idle, EDF</td>
</tr>
<tr>
<td>LDF</td>
<td>EDF with modifications</td>
<td>Search</td>
</tr>
</tbody>
</table>

Homework: due Friday April 5 in class

- Read chapters 1, 2, 3 of the Buttazzo book.
- [20 points] Ex. 3.1 from [Buttazzo]: check whether EDD produces a feasible schedule for the task set:
  P1: $[C = 4, \ d = 9]$
  P2: $[C = 5, \ d = 16]$
  P3: $[C = 2, \ d = 5]$
  P4: $[C = 3, \ d = 10]$

Announcements

- Midterms:
  - Friday April 19
  - Friday May 10
  (in class)

Homework (cont.)

- [40 points] Consider two sets of processes $[[C_1^1, d_1^1], [C_2^1, d_2^1], \ldots, [C_m^1, d_m^1]]$
  $[[C_1^2, d_1^2], [C_2^2, d_2^2], \ldots, [C_n^2, d_n^2]]$
  and assume that they are EDD-schedulable. Prove that their union is also EDD-schedulable on a processor that is twice as fast. Note: a precise proof is required, not a common-sense argument.

Homework (cont.)

- Ex. 3.5 of Buttazzo: given seven tasks $A,B,C,D,E,F,G$, construct the precedence graph from the constraints:
  $A < C, \ B < C, \ C < E, \ D < F, \ B < D, \ C < F, \ D < G$
- [40 points] Assuming that all tasks arrive at time 0, have deadline 20 (all of them), and have computation times $C$ equal to $2,3,5,1,2,5$ respectively, modify their arrival times and deadlines to schedule them with EDF.