CIS 721 - Real-Time Systems
Lecture 4: On-Line Scheduling

Mitch Neilsen
neilsen@cis.ksu.edu

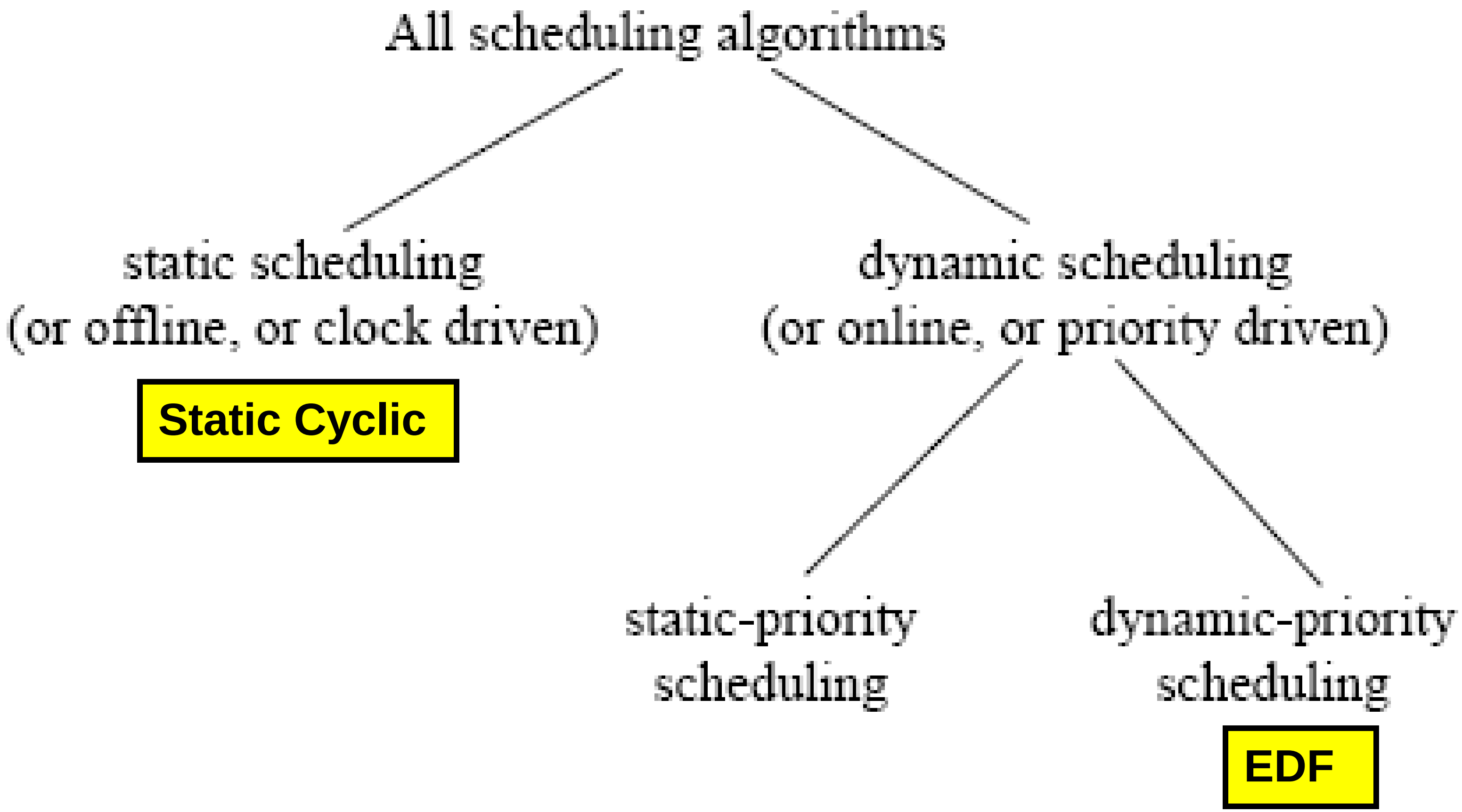
Outline

- **Approaches For Real-Time Scheduling (Ch. 4)**
  - **Off-line Scheduling**
  - **On-line Scheduling**
    - **EDF**
    - **Competitive Factor Theorem**
- **Clock-Driven Scheduling (Ch. 5)**
- Priority-Driven Scheduling (Ch.6-7)

Schedules

- A **schedule** is an assignment of jobs to available processors. In a **feasible schedule**, every job starts at or after its release time and completes by its deadline in a **hard** real-time system.
- A scheduling algorithm is **optimal** if it always produces a feasible schedule if such a schedule exists.

Classification of Scheduling Algorithms



EDF is optimal for scheduling preemptive tasks on a single processor.

Temporal Parameters

- J_i : **job** – a unit of work
- T_i (or τ_i): **task** - a set of related jobs
- A **periodic task** is sequence of invocations of jobs with identical parameters.
- r_i: **release time** of job J_i
- d_i: **absolute deadline** of job J_i
- D_i: **relative deadline** (or just **deadline**) of job J_i
- e_i: (Maximum) **execution time** of job J_i

Periodic Task Model

- **Tasks:** T_1, ….., T_n
- Each consists of a set of **jobs**: $T_i = \{J_{i1}, J_{i2}, \ldots\ldots\}$
- ϕ_i: p**hase** of task T_i = time when its first job is released
- p_i: p**eriod** of T_i = minimum inter-release time
- H: h**yperperiod** $H = \mathrm{lcm}(p_1, \ldots.., p_n)$
- e_i: e**xecution time** of T_i
- u_i: **utilization** of task T_i is given by $u_i = e_i / p_i$
- D_i: (relative) **deadline** of T_i, typically $D_i = p_i$

Periodic Task

- We refer to a periodic task T_i with phase ϕ_i , period p_i , execution time e_i , and relative deadline D_i by the 4-tuple (ϕ_i , p_i , e_i , D_i).
- Example: (1, 10, 3, 6)
- By default, the phase of each task is 0, and its relative deadline is equal to its period.
- Example: (0, 10, 3, 10) = (10, 3).

Sample Periodic Task Set

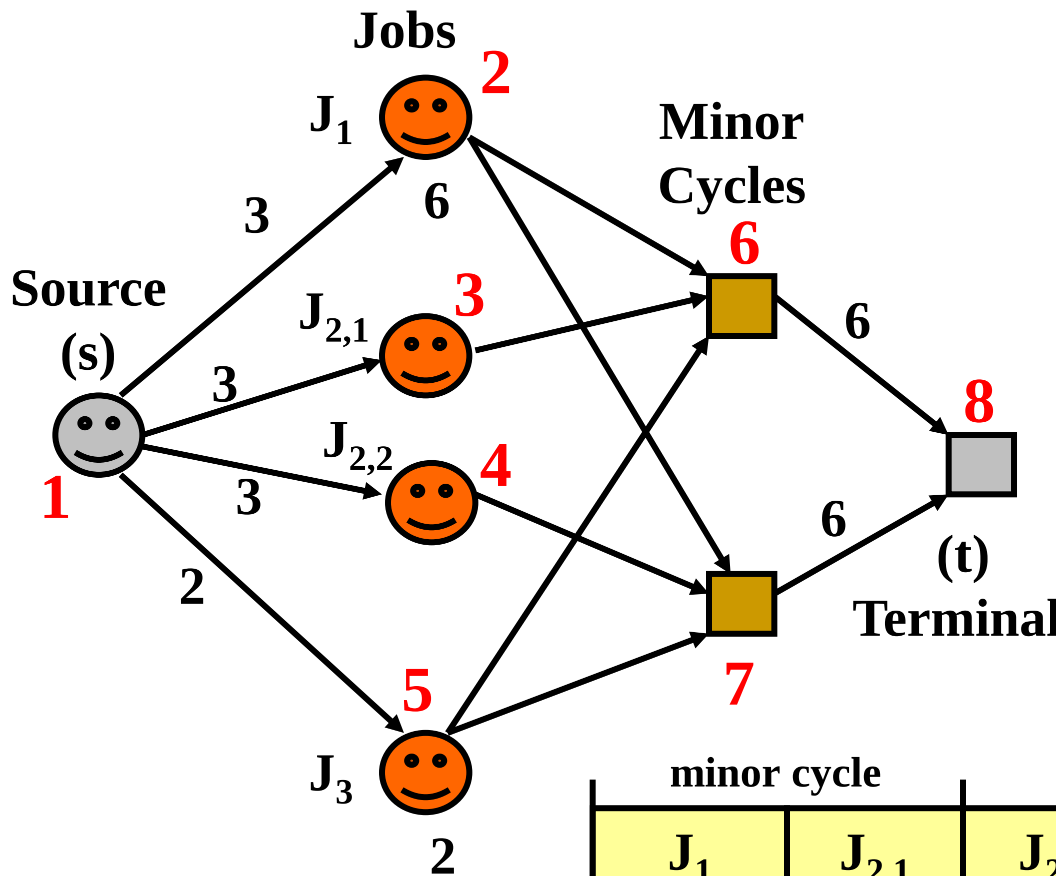
- Task 1: (12, 3)
- Task 2: (6, 3)
- Task 3: (12, 2)

- Hyperperiod H = lcm(12, 6, 12) = 12
- Potential frame sizes? 3 or 6

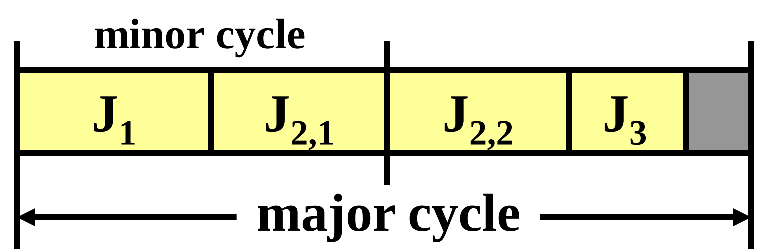
Clock-Driven Example

$ hi_pr < cyclic2.input > cyclic2.output

INPUT:
p max 8 12
n 1 s
n 8 t
a 1 2 3
a 1 3 3
a 1 4 3
a 1 5 2
a 2 6 6
a 2 7 6
a 3 6 6
a 4 7 6
a 5 6 6
a 5 7 6
a 6 8 6
a 7 8 6



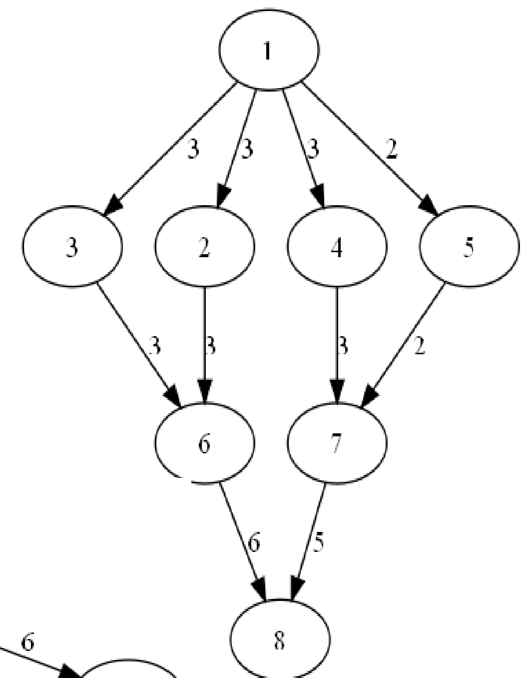
OUTPUT:
max flow:11
c flow values
f 1 2 3
f 1 4 3
f 1 3 3
f 1 5 2
f 2 6 3
f 2 7 0
f 3 6 3
f 4 7 3
f 5 7 2
f 5 6 0
f 6 8 6
f 7 8 5
c

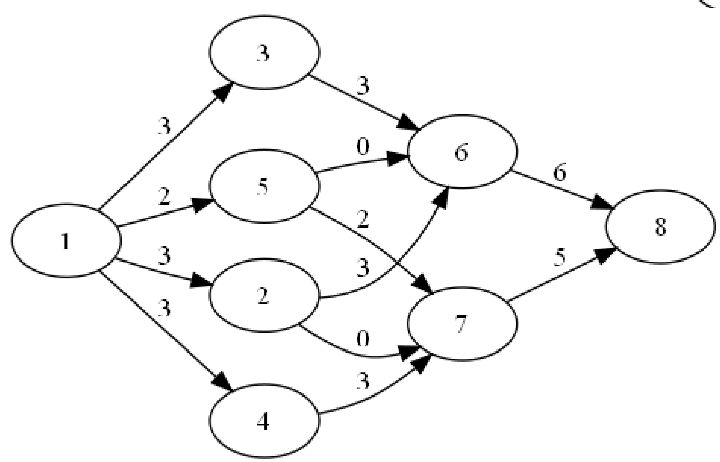


Resulting Graphics File

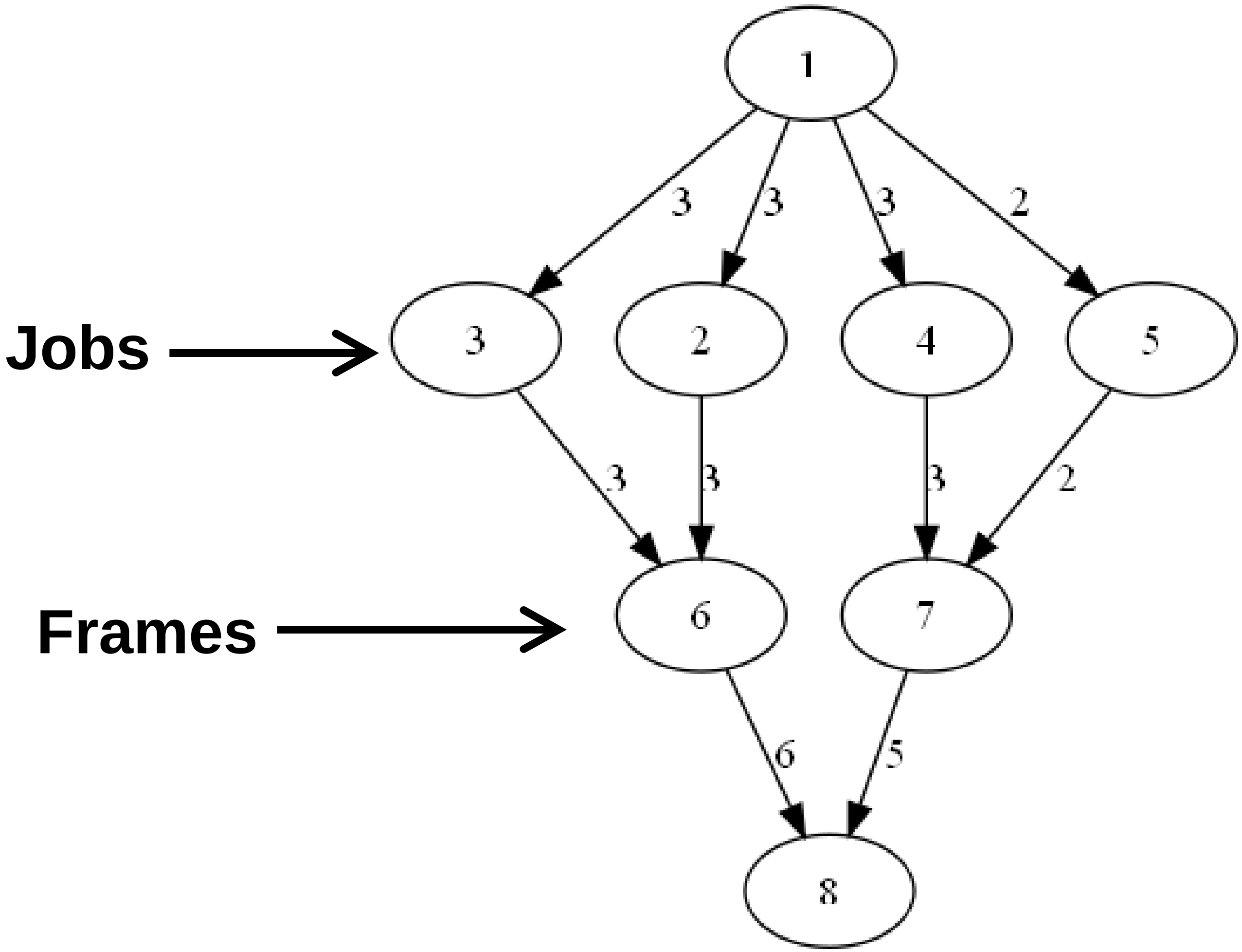
```
digraph g {
1 -> 2 [label=3]
1 -> 4 [label=3]
1 -> 3 [label=3]
1 -> 5 [label=2]
2 -> 6 [label=3]
3 -> 6 [label=3]
4 -> 7 [label=3]
5 -> 7 [label=2]
6 -> 8 [label=6]
7 -> 8 [label=5]
{ rank=same;  2; 3; 4; 5;}
{ rank=same;  6; 7;}
}
```

`dot -Tpng fn.gr -o fn.png`

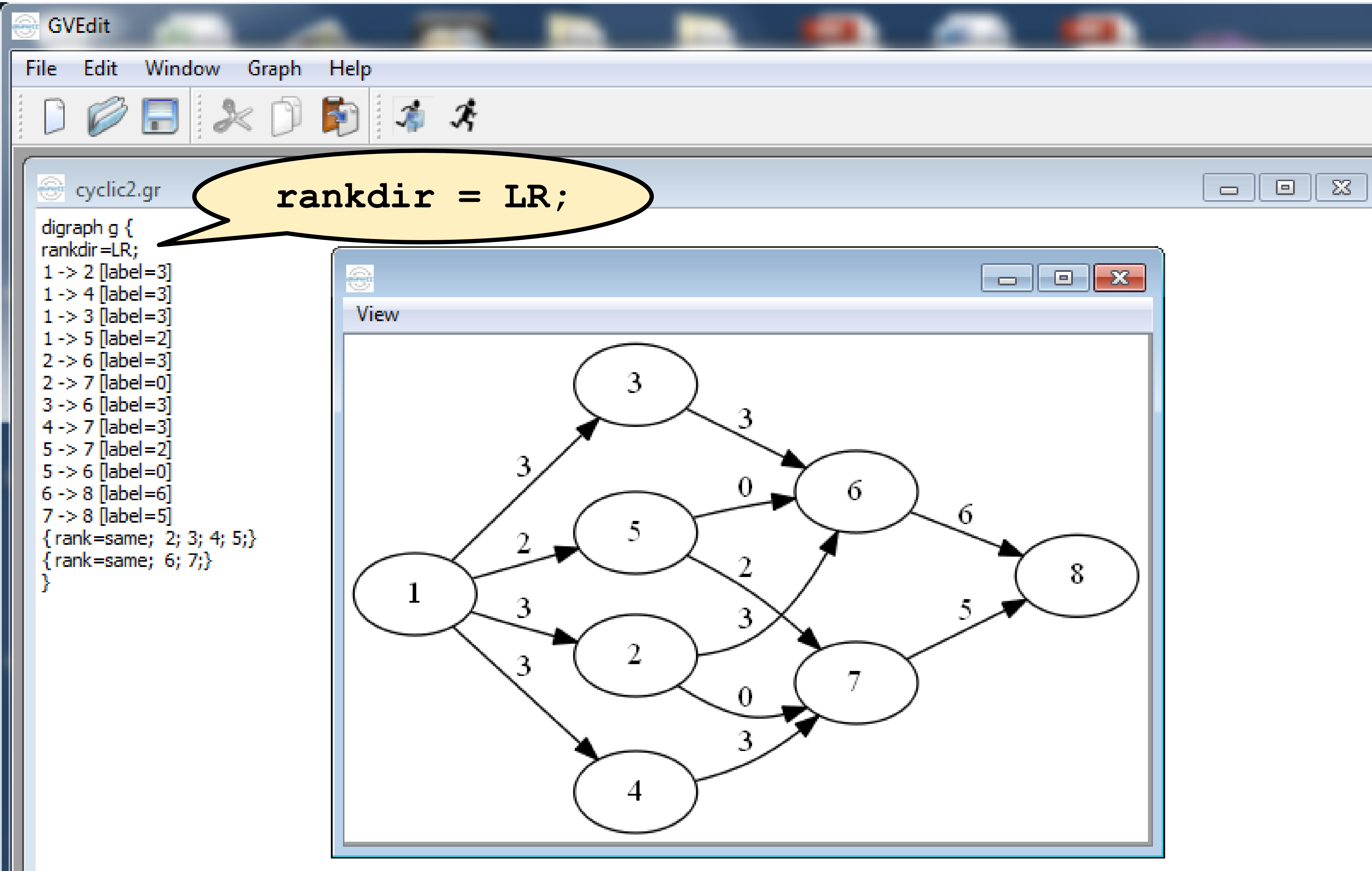




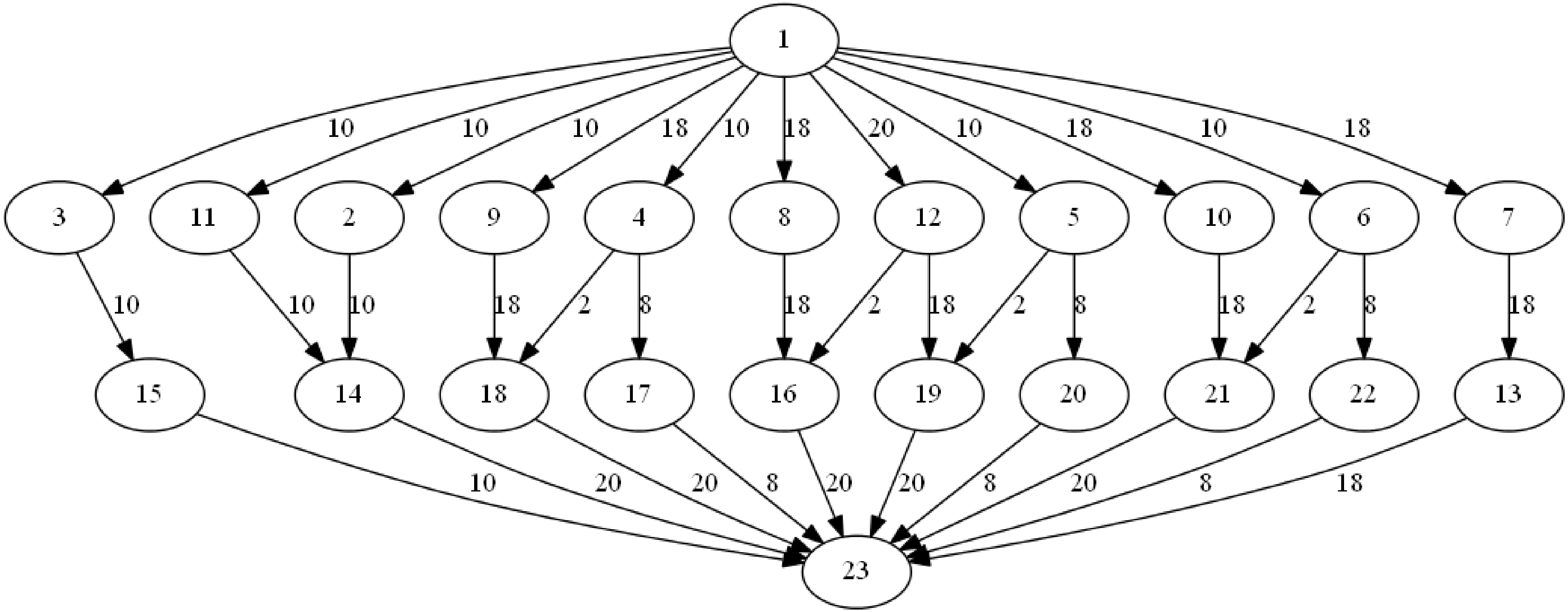
Output generated



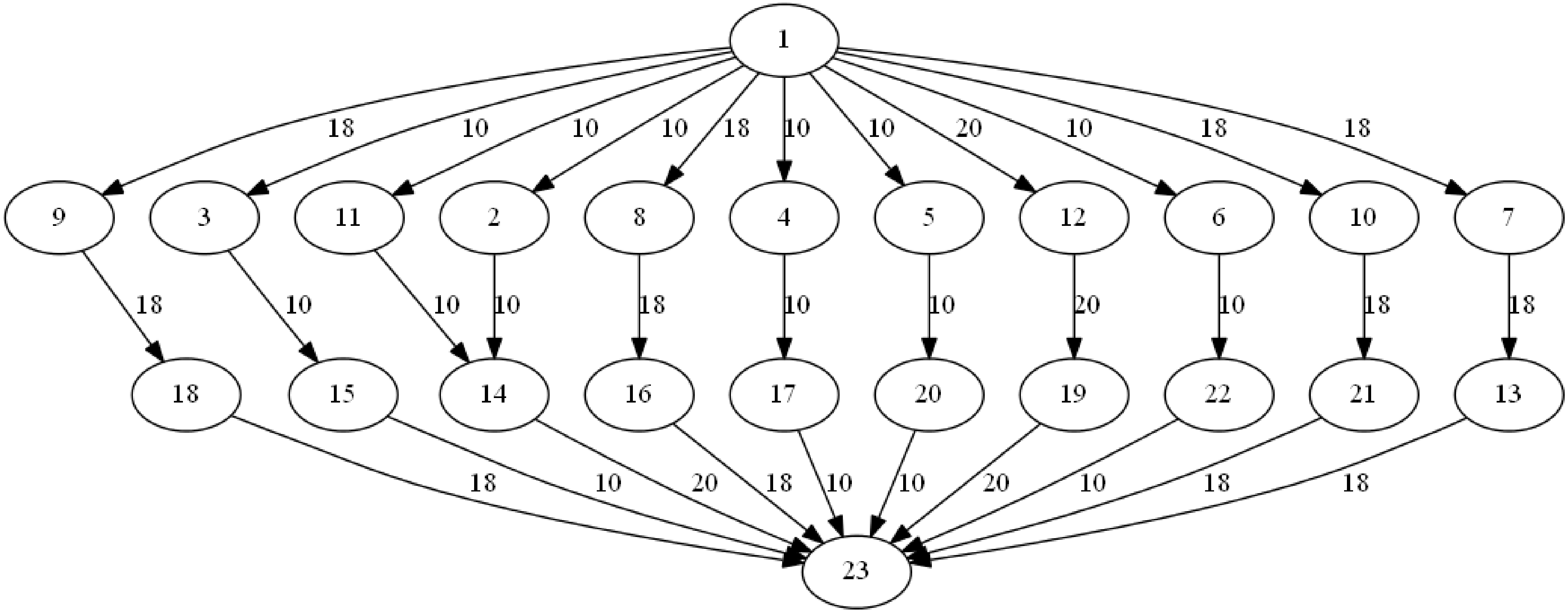
GraphViz Editor: GVEdit



Another Example



Another Example - Constrained



On-line Scheduling

- Scheduling is performed **on-line** if the scheduler makes decisions without knowledge of the jobs to be released in the future.
- EDF is an example of an on-line scheduling algorithm.
- The system is **overloaded** if the jobs cannot be feasibly scheduled even by a clairvoyant scheduler. During an overload, some jobs must be discarded in order to allow other jobs to complete.

EDF Algorithm

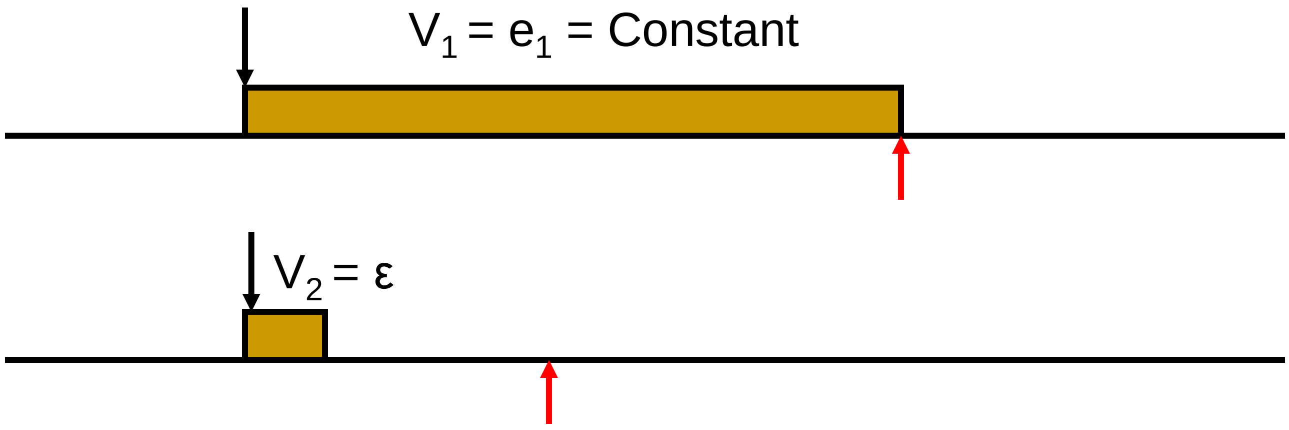
- **Earliest-Deadline-First (EDF) algorithm:**
  - At any time, execute the available job with the **earliest deadline**.
- **Theorem: (Optimality of EDF):** In a system with **one processor** and **preemption** allowed, EDF is optimal; that is, EDF can produce a feasible schedule for a given job set J with arbitrary release times and deadlines, if a feasible schedule exists.
- **Proof:** Apply schedule transformations and remove idle time.

Competitive Factor

- For hard real-time systems, the **value of a job** is defined to be the execution time of the job if the job completes by its deadline.
- The **value of a schedule** of a sequence of jobs is the sum of the values of all jobs in the sequence.
- An on-line algorithm has a **competitive factor *c*** iff the value of the schedule of any finite sequence of jobs produced by the algorithm is at least *c* times the value of the schedule produced by an optimal, clairvoyant algorithm.

Competitive Factor of EDF

- The competitive factor of EDF is 0.

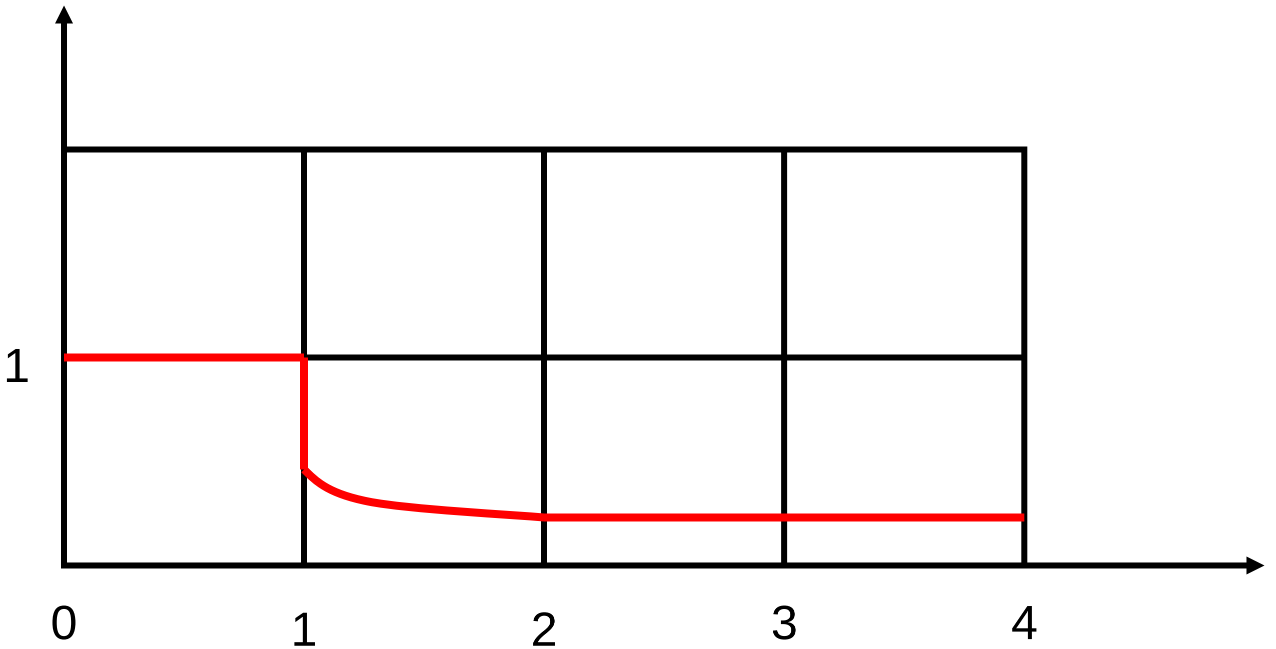


The value of an EDF schedule is V_2 whereas the value of an optimal schedule is V_1. The competitive factor of EDF = V_2 / V_1 approaches 0 as ε approaches 0.

On-line Scheduling Limitations – Or how bad is it?

- **Theorem:** No on-line scheduling algorithm can achieve a competitive factor greater than 0.25 when the system is overloaded.
- Proof: [Baruah, et al., 1991] Competitive_Factor_Theorem.pdf
- Note: Even when the system is only slightly overloaded (e.g., slightly more than 100% utilization), no on-line algorithm can achieve a competitive factor greater than 0.385.

Maximum Competitive Factor



Summary

- Read Ch. 4-6.
- Homework #1.