technische universität
dortmund

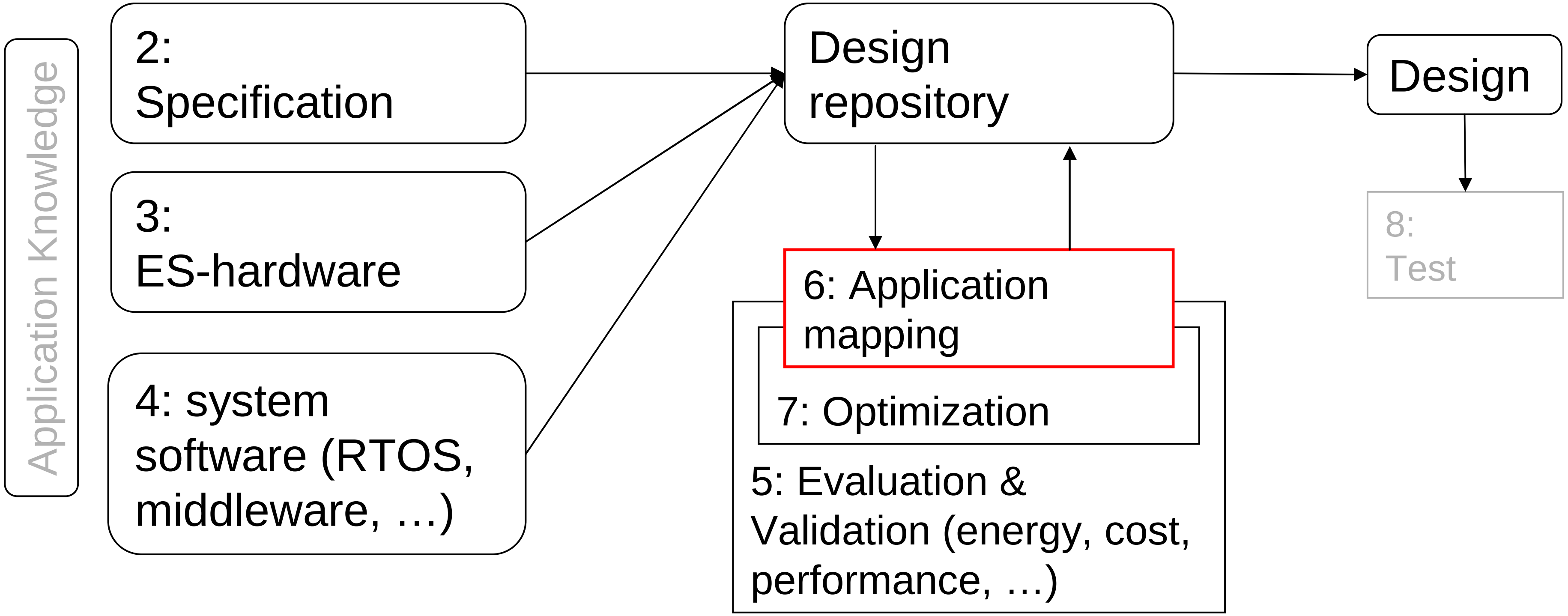
fakultät für informatik
informatik 12

Mapping of Applications to Platforms

Peter Marwedel
TU Dortmund, Informatik 12
Germany

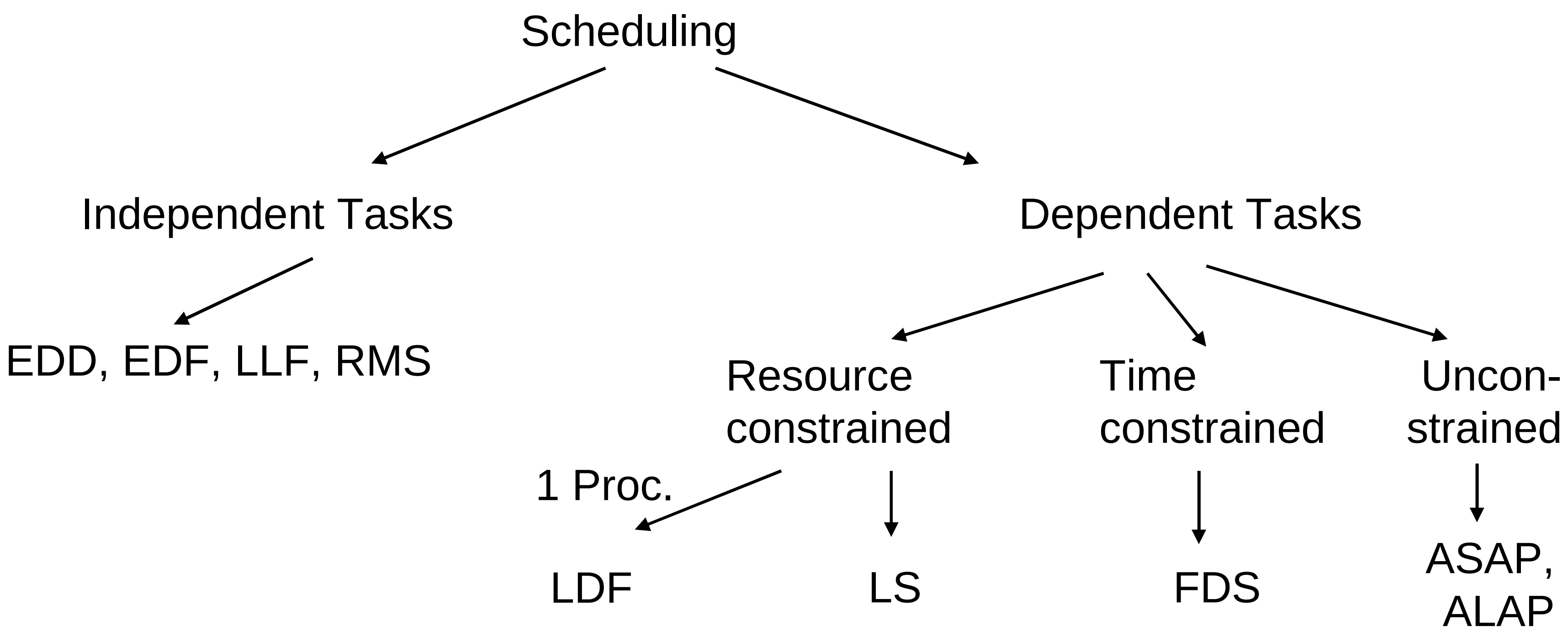
2010年 12 月 10 日

Structure of this course



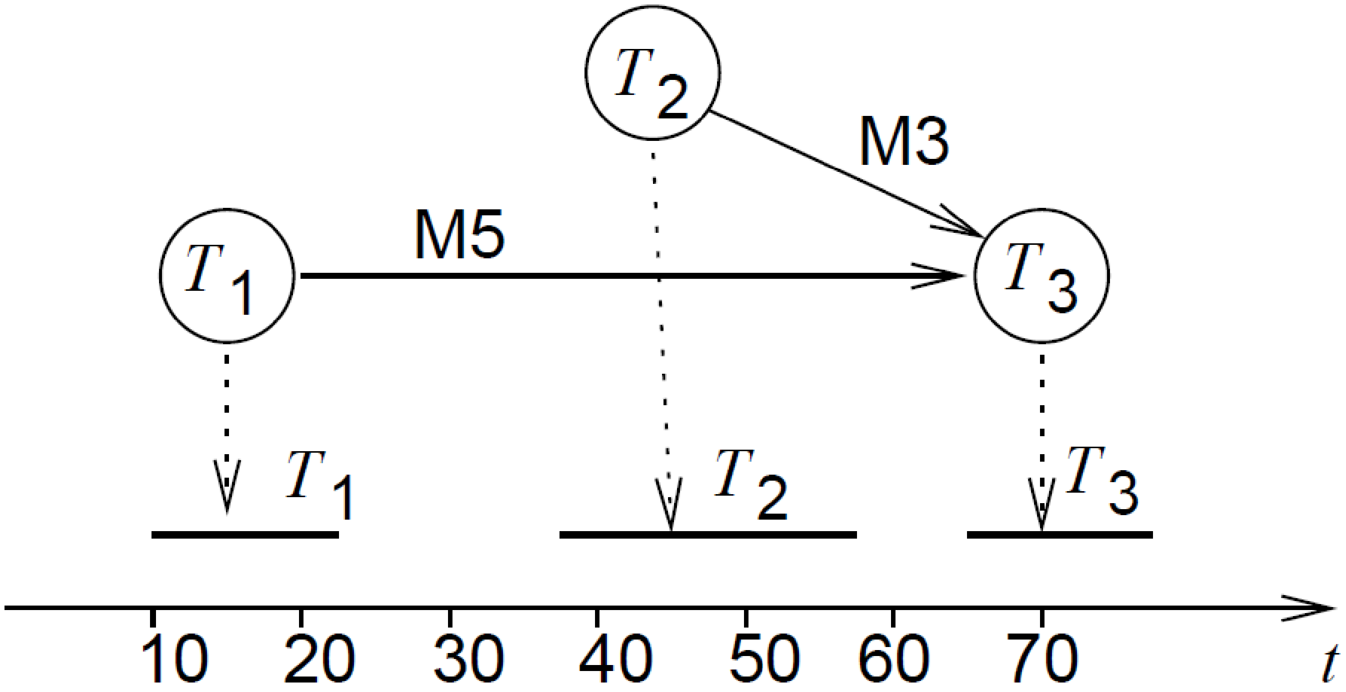
Numbers denote sequence of chapters

Classification of Scheduling Problems



Scheduling with precedence constraints

Task graph and possible schedule:

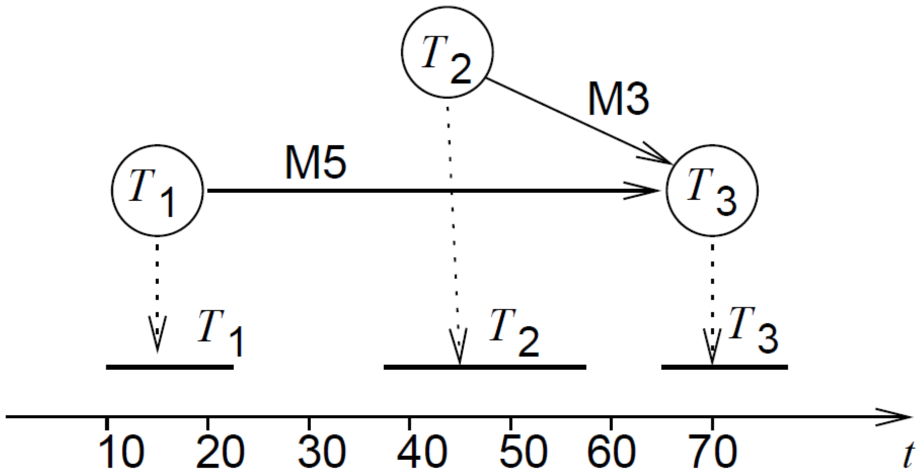


Simultaneous Arrival Times: The Latest Deadline First (LDF) Algorithm

LDF [Lawler, 1973]: reads the task graph and among the tasks with no successors inserts the one with the latest deadline into a queue. It then repeats this process, putting tasks whose successor have all been selected into the queue.

At run-time, the tasks are executed in the generated total order.

LDF is non-preemptive and is optimal for mono-processors.



If no local deadlines exist, LDF performs just a topological sort.

Asynchronous Arrival Times: Modified EDF Algorithm

This case can be handled with a modified EDF algorithm. The key idea is to transform the problem from a given set of dependent tasks into a set of independent tasks with different timing parameters [Chetto90].

This algorithm is optimal for mono-processor systems.

If preemption is not allowed, the heuristic algorithm developed by Stankovic and Ramamritham can be used.

Dependent tasks

The problem of deciding whether or not a schedule exists for a set of dependent tasks and a given deadline is NP-complete in general [Garey/Johnson].

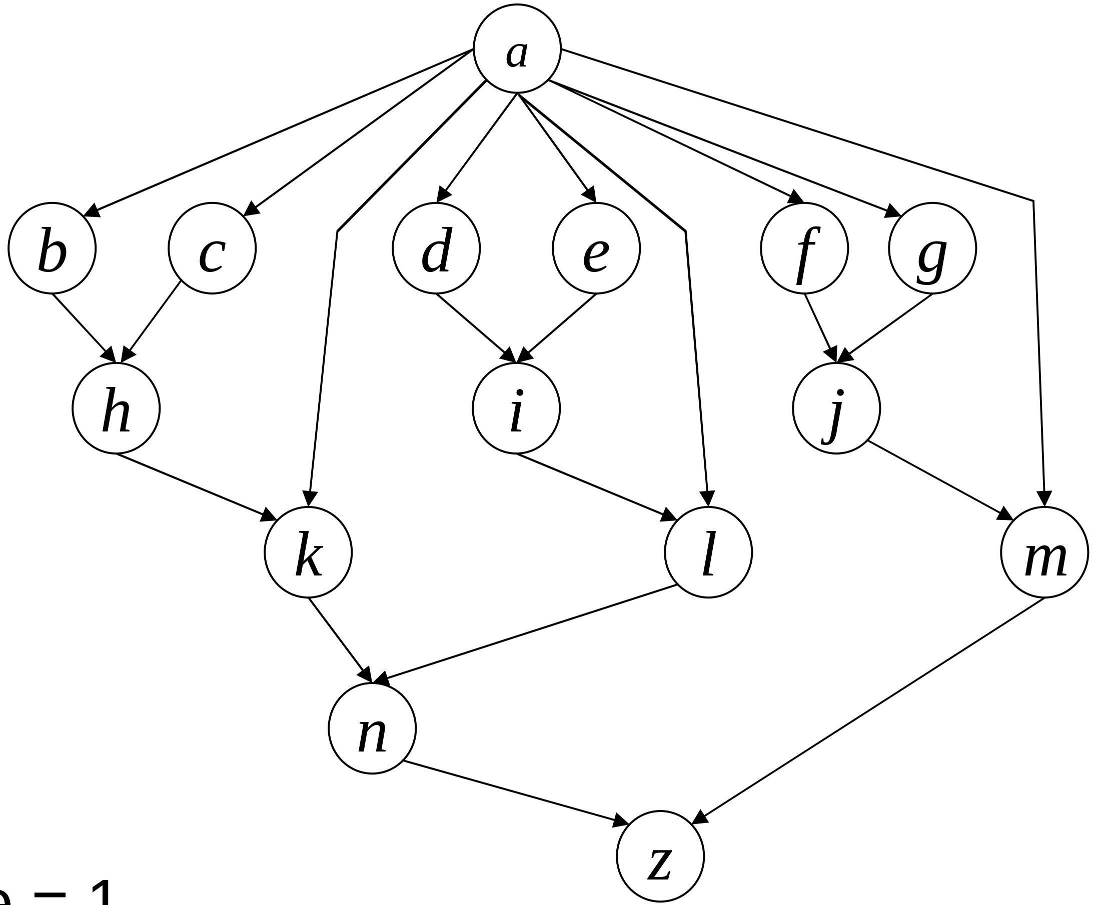
Strategies:

1. Add resources, so that scheduling becomes easier
2. Split problem into static and dynamic part so that only a minimum of decisions need to be taken at run-time.
3. ➡ Use scheduling algorithms from high-level synthesis

Classes of mapping algorithms considered in this course

- **Classical scheduling algorithms**
  Mostly for independent tasks & ignoring communication, mostly for mono- and homogeneous multiprocessors
- **Dependent tasks as considered in architectural synthesis**
  Initially designed in different context, but applicable
- **Hardware/software partitioning**
  Dependent tasks, heterogeneous systems, focus on resource assignment
- **Design space exploration using genetic algorithms**
  Heterogeneous systems, incl. communication modeling

Task graph



Assumption:
execution time = 1
for all tasks

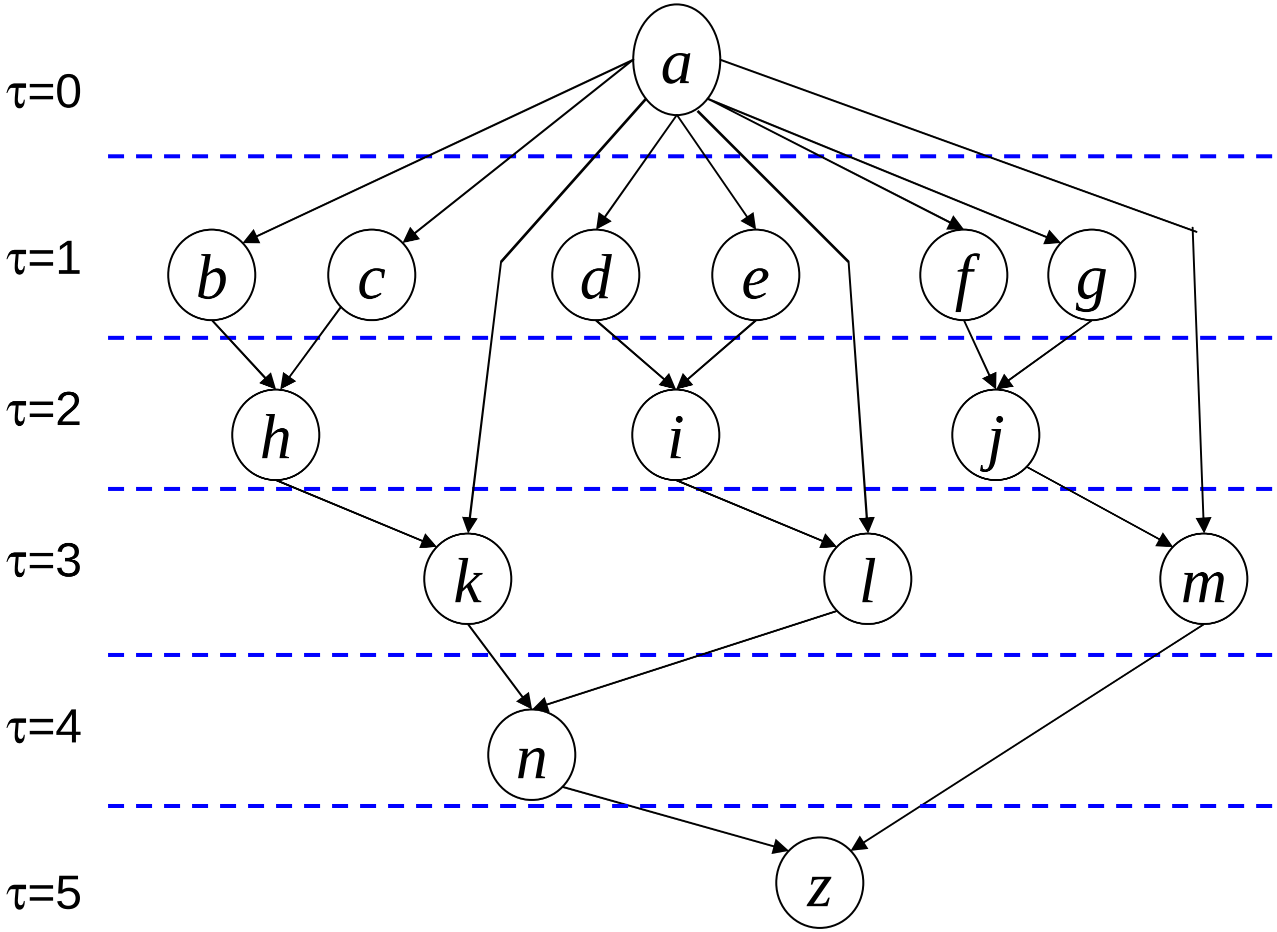
As soon as possible (ASAP) scheduling

ASAP: All tasks are scheduled as early as possible

Loop over (integer) time steps:

- Compute the set of unscheduled tasks for which all predecessors have finished their computation
- Schedule these tasks to start at the current time step.

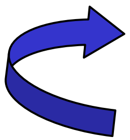
As soon as possible (ASAP) scheduling: Example



As-late-as-possible (ALAP) scheduling

ALAP: All tasks are scheduled as late as possible

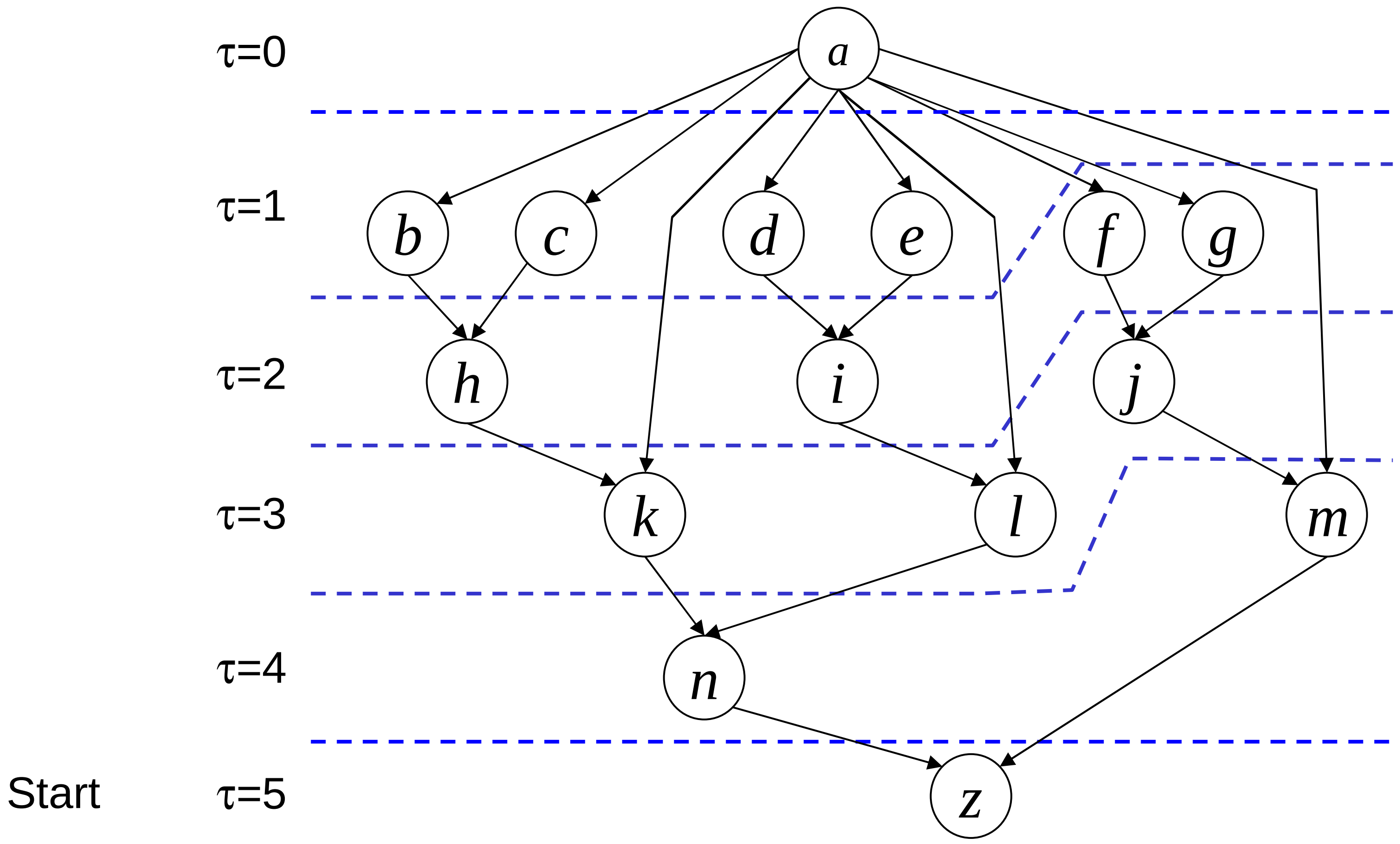
Start at last time step*:



Schedule tasks with no successors and tasks for which all successors have already been scheduled.

* Generate a list, starting at its end

As-late-as-possible (ALAP) scheduling: Example



(Resource constrained) List Scheduling

List scheduling: extension of ALAP/ASAP method

Preparation:

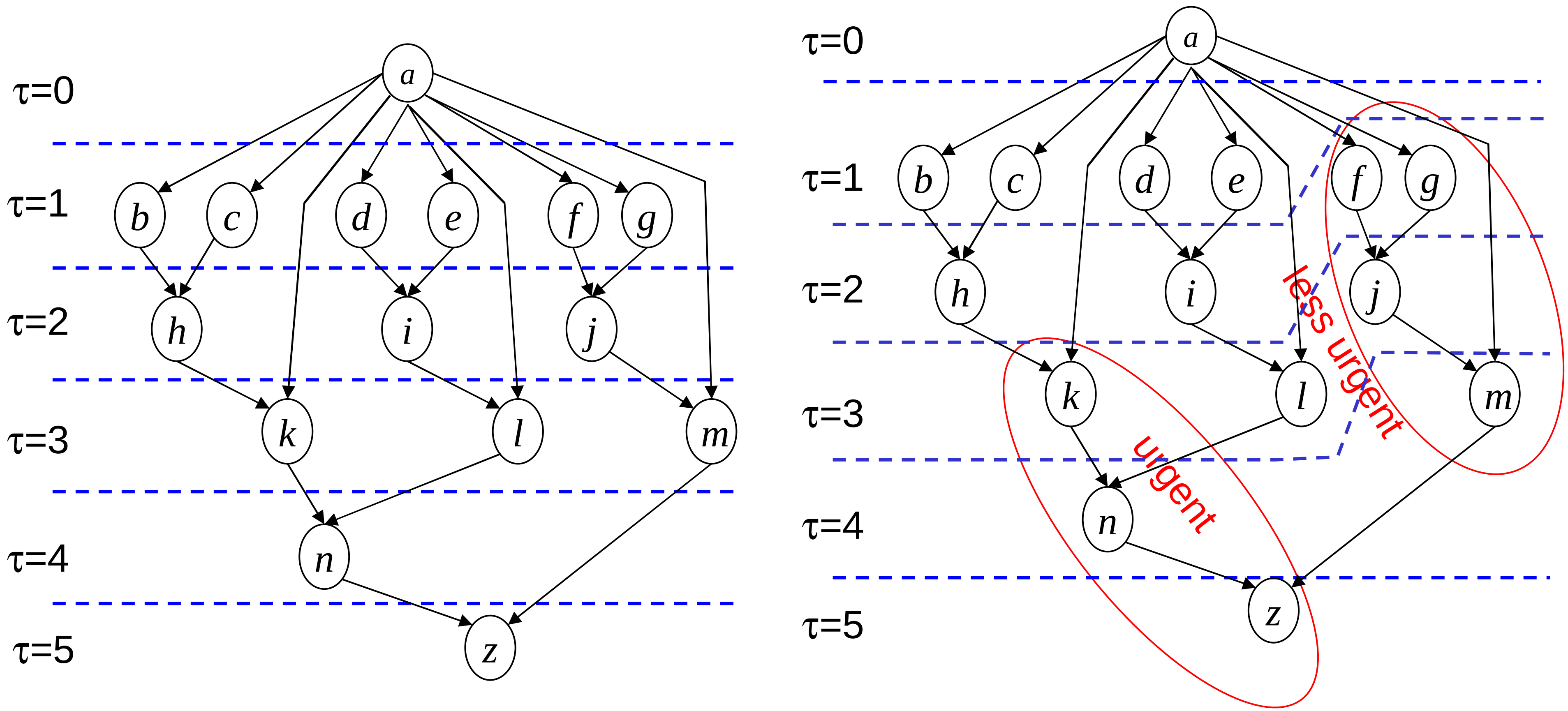
- Topological sort of task graph $G=(V,E)$
- Computation of priority of each task:

  Possible priorities u:

  - Number of successors
  - Longest path
  - **Mobility** = τ (ALAP schedule)- τ (ASAP schedule)

Mobility as a priority function

Mobility is not very precise



Algorithm

List($G(V,E)$, B, u){
i :=0;
repeat {
Compute set of candidate tasks A_i ;
Compute set of not terminated tasks G_i ;
Select $S_i \subseteq A_i$ of maximum priority r such that
$| S_i | + | G_i | \leq B$ (*resource constraint*)
foreach ($v_j \in S_i$): $\tau(v_j)$:=i; (*set start time*)
i := i +1;
}
until (all nodes are scheduled);
return (τ);
}

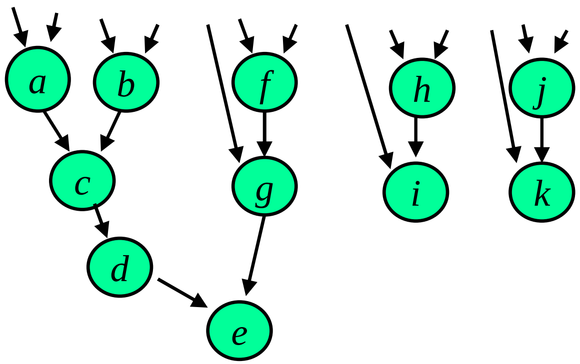
} may be repeated for different task/ processor classes

Complexity: $O(|V|)$

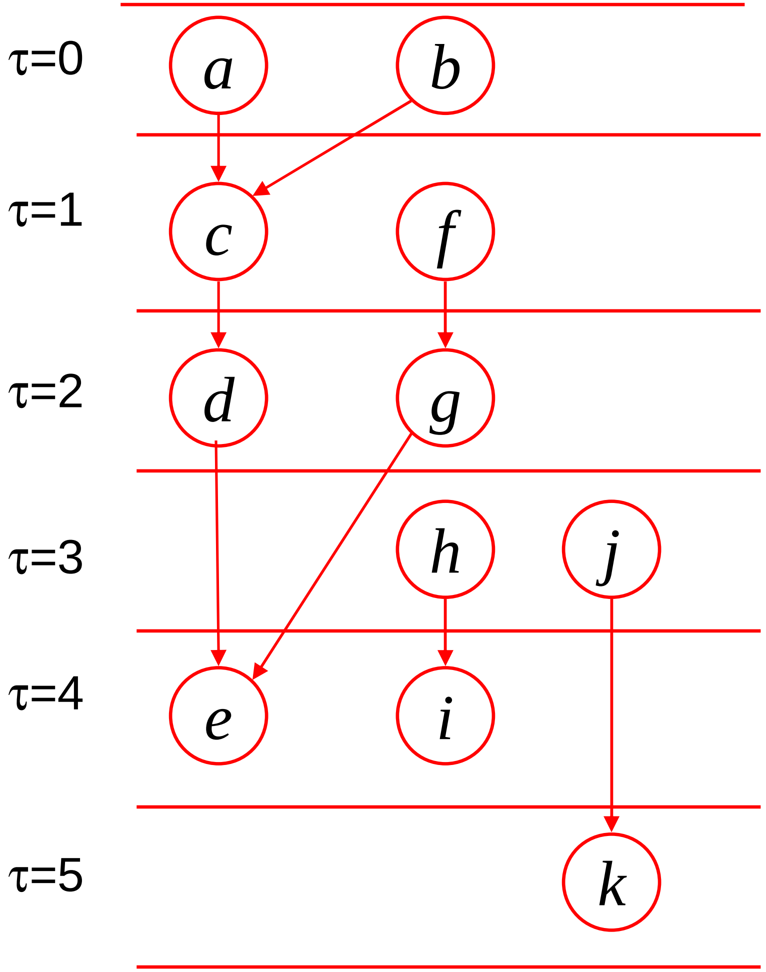
Example

Assuming B =2, unit execution time and u : path length

$u(a)= u(b)=4$
$u(c)= u(f)=3$
$u(d)= u(g)= u(h)= u(j)=2$
$u(e)= u(i)= u(k)=1$
$\forall\ i : G_i =0$

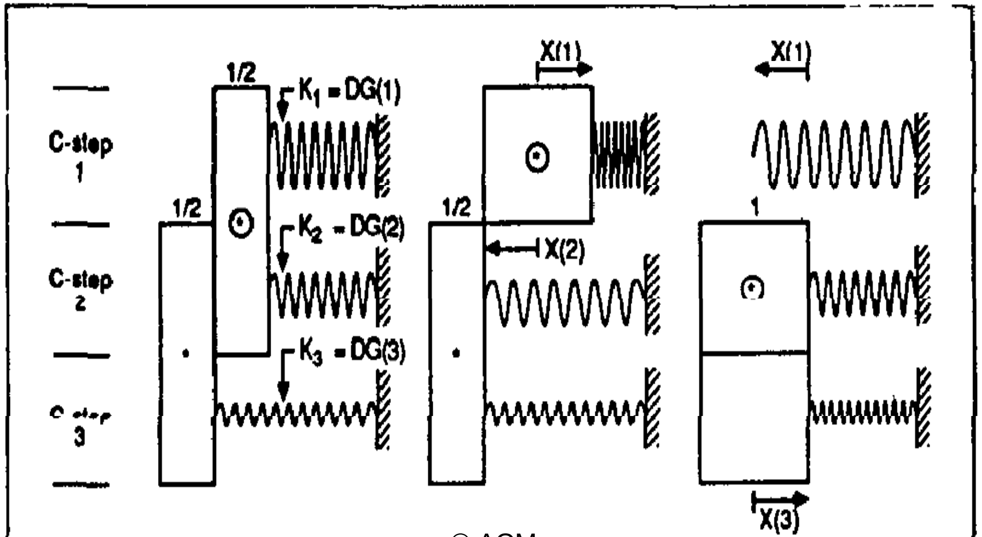


Modified example based on J. Teich



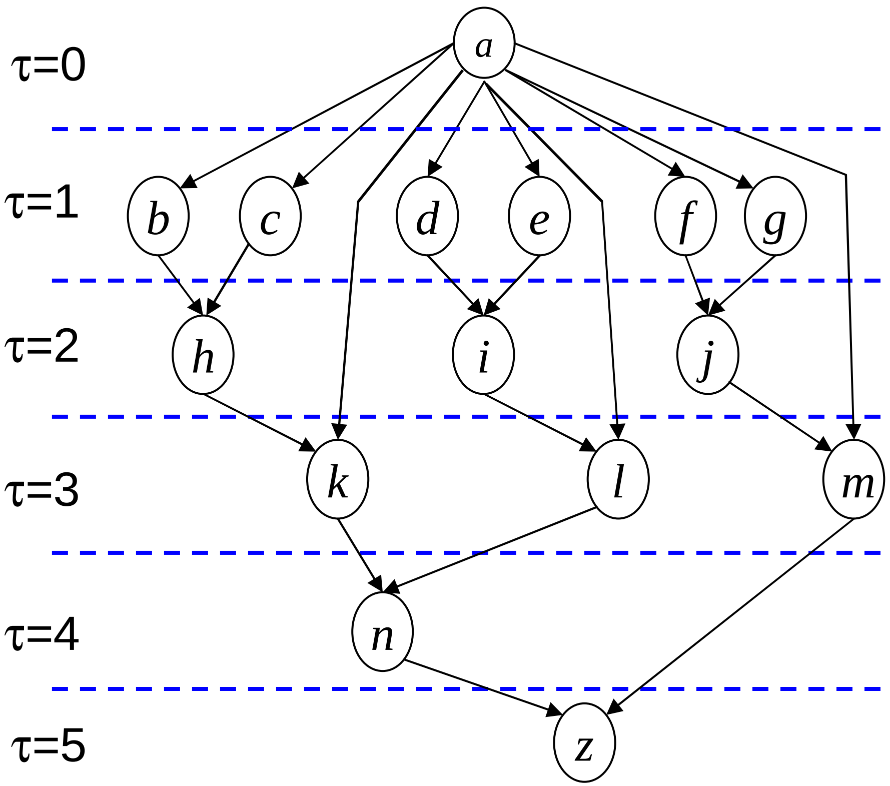
(Time constrained) Force-directed scheduling

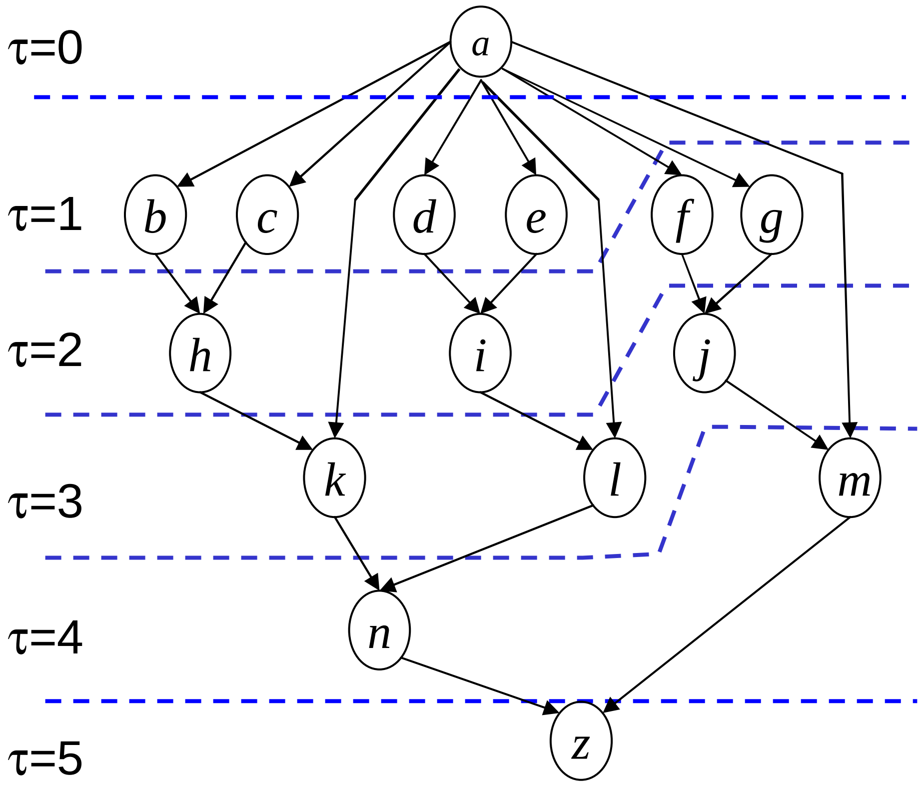
Goal: balanced utilization of resources
Based on spring model;
Originally proposed for high-level synthesis



* [Pierre G. Paulin, J.P. Knight, Force-directed scheduling in automatic data path synthesis, *Design Automation Conference* (DAC), 1987, S. 195-202]

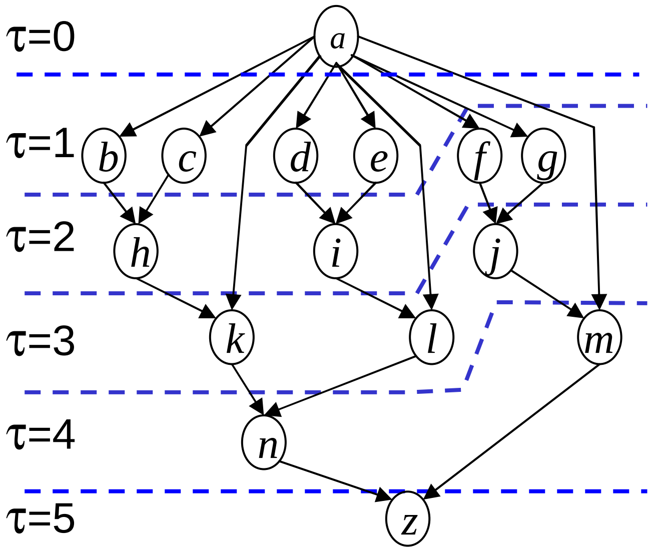
Phase 1: Generation of ASAP and ALAP Schedule

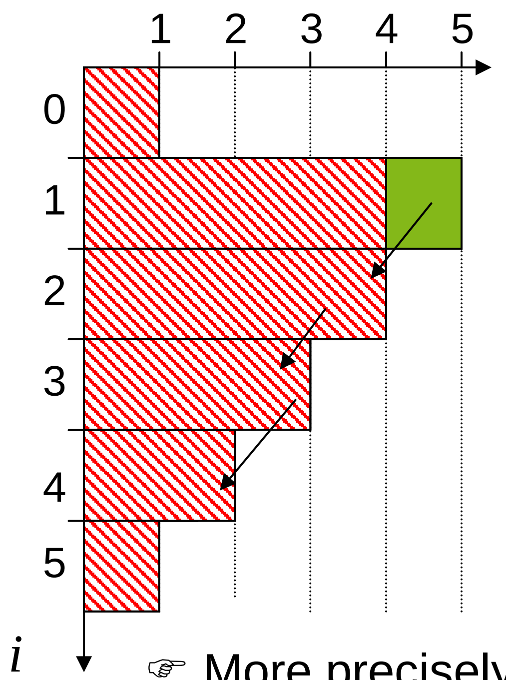




Next: computation of "forces"

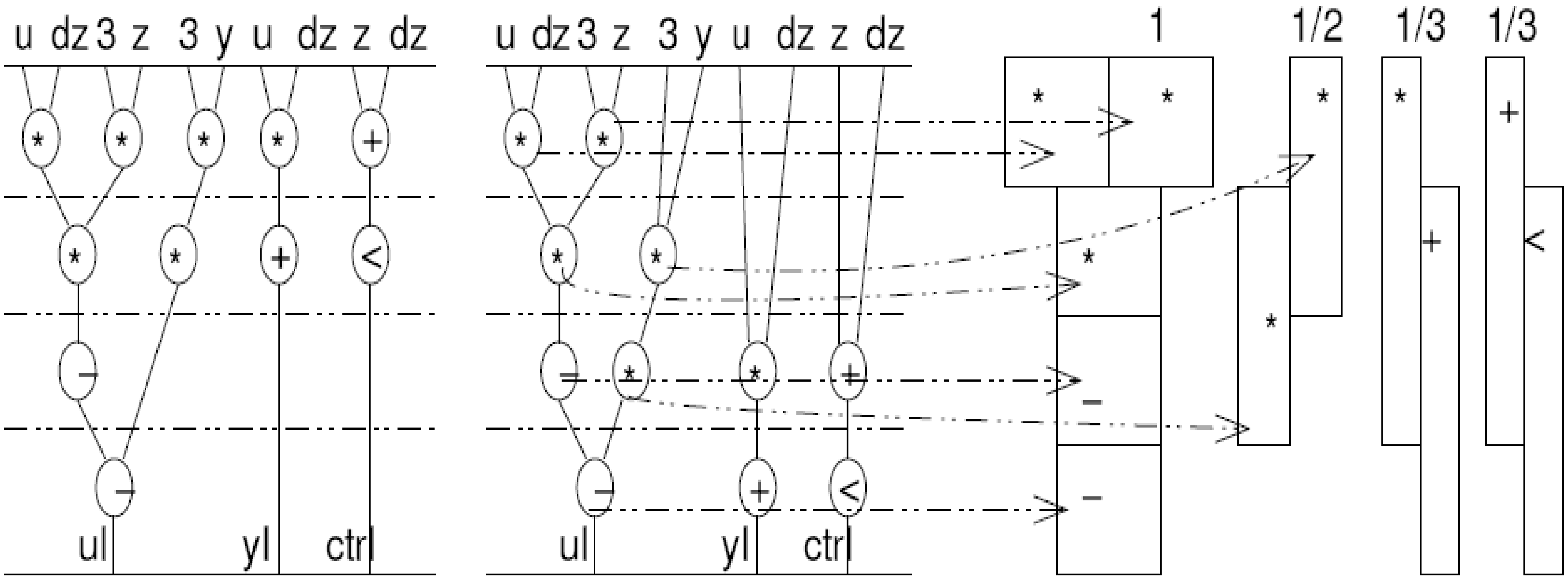
- Direct forces push each task into the direction of lower values of $D(i)$.
- Impact of direct forces on dependent tasks taken into account by indirect forces
- Balanced resource usage ≈ smallest forces
- For our simple example and time constraint=6: result = ALAP schedule





☞ More precisely …

1.Compute time frames $R(j)$
2. Compute “probability“ $P(j,i)$ of assignment $j \rightarrow i$

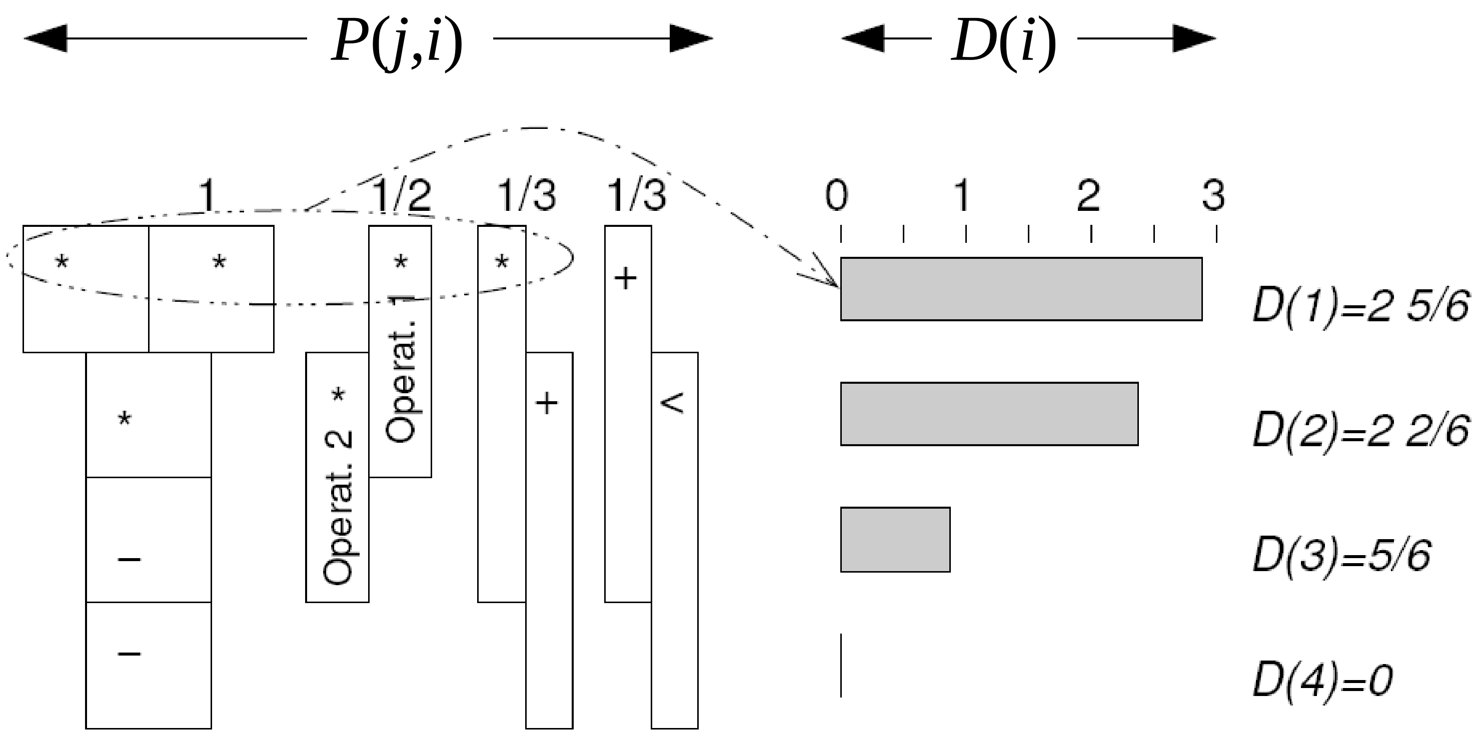


$R(j)$={ASAP-control step … ALAP-control step}

$$P(j,i) = \begin{cases} \frac{1}{|R(j)|} & \text{if} \quad i \in R(j) \\ 0 & \text{otherwise} \end{cases}$$

3. Compute "distribution" *D*(*i*) (# Operations in control step *i*)

$$D(i) = \sum_{j,type(j)\in H} P(j,i)$$



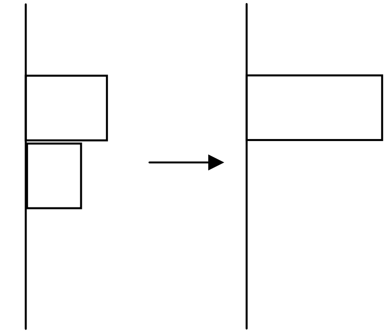
4. Compute direct forces (1)

- $\Delta P_i(j,i')$: Δ for force on task j in time step i',

  if j is mapped to time step i.

  The new probability for executing j in i is 1;
  the previous was $P(j, i)$.

  The new probability for executing j in $i' \neq i$ is 0;
  the previous was $P(j, i)$.



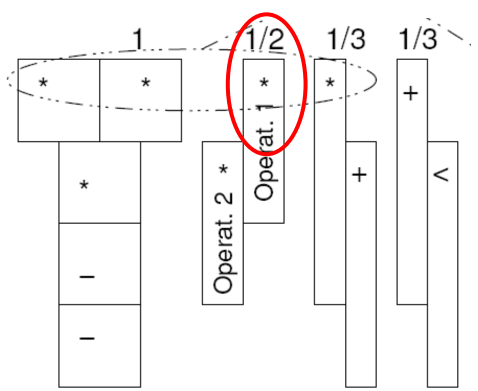
☞ $$\Delta P_i(j,i') = \begin{cases} 1 - P(j,i) & \text{if} \quad i = i' \\ -P(j,i') & \text{otherwise} \end{cases}$$

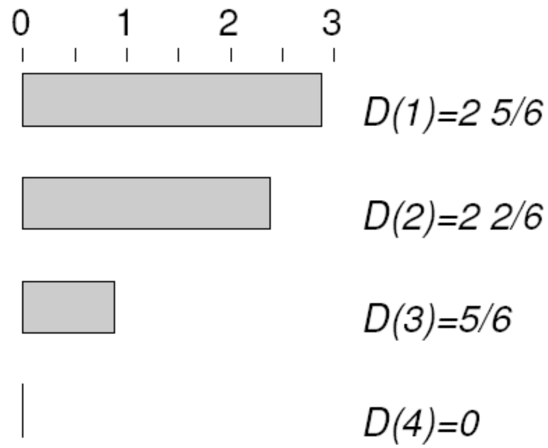
4. Compute direct forces (2)

- $SF(j, i)$ is the overall change of direct forces resulting from the mapping of j to time step i.

$$SF(j,i) = \sum_{i' \in R(j)} D(i')\Delta P_i(j,i') \qquad \Delta P_i(j,i') = \begin{cases} 1 - P(j,i) & \text{if} \quad i = i' \\ -P(j,i') & \text{otherwise} \end{cases}$$

Example



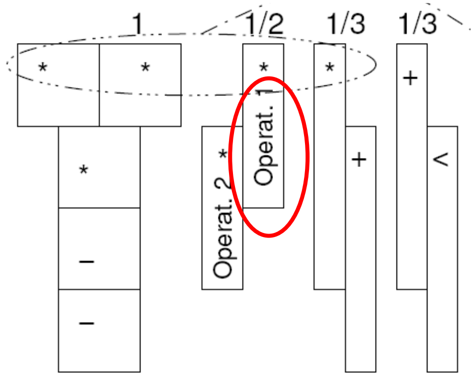


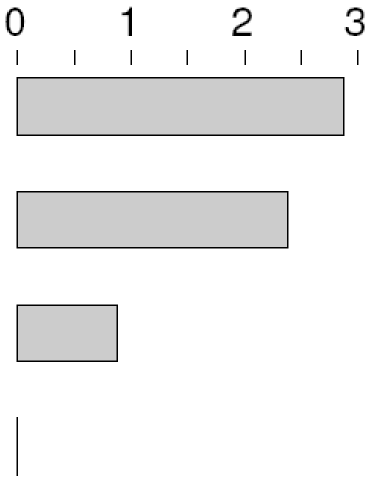
$SF(1,1)=2\ 5/6\ (1–1/2) – 2\ 2/6\ (1/2)=$

$1/2\ (17/6–14/6) = 1/2\ (3/6) = 1/4$

4. Compute direct forces (3)

Direct force if task/operation 1 is mapped to time step 2





D(1)=2 5/6

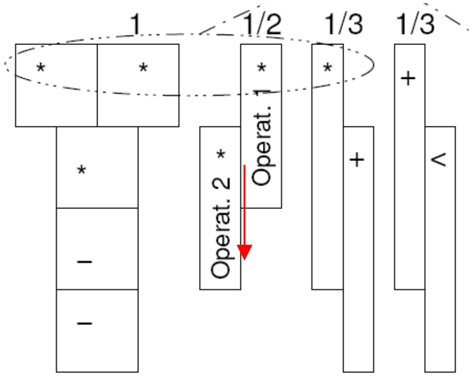
D(2)=2 2/6

D(3)=5/6

D(4)=0

$$
\begin{aligned}
SF(1,2) &= D(1) * \Delta P_2(1,1) + D(2) * \Delta P_2(1,2) \\
&= 2\frac{5}{6} * (-0,5) + 2\frac{2}{6} * 0.5 \\
&= -\frac{17}{12} + \frac{14}{12} \\
&= -\frac{3}{12} = -\frac{1}{4}
\end{aligned}
$$

5. Compute indirect forces (1)



Mapping task 1 to time step 2 implies mapping task 2 to time step 3

☞ Consider predecessor and successor forces:

$$VF(j,i) = \sum_{j' \in \text{ predecessor of } j} \; \sum_{i' \in I} D(i')\Delta P_{j,i}(j',i')$$

$$NF(j,i) = \sum_{j' \in \text{ successor of } j} \; \sum_{i' \in I} D(i')\Delta P_{j,i}(j',i')$$

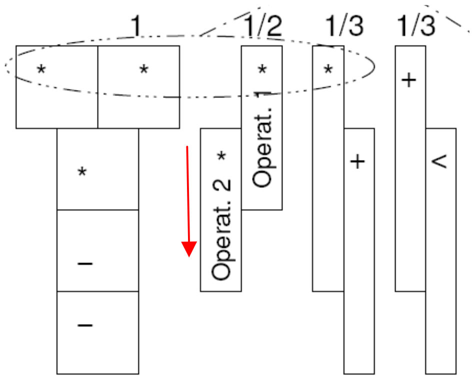
$\Delta P_{j,i}(j',i')$ is the Δ in the probability of mapping j' to i' resulting from the mapping of j to i

5. Compute indirect forces (2)

$$VF(j,i) = \sum_{j' \in \text{predecessor of } j} \sum_{i' \in I} D(i')\Delta P_{j,i}(j',i')$$

$$NF(j,i) = \sum_{j' \in \text{successor of } j} \sum_{i' \in I} D(i')\Delta P_{j,i}(j',i')$$

Example: Computation of successor forces for task 1 in time step 2



$$\begin{aligned} NF(1,2) &= D(2) * \Delta P_{1,2}(2,2) + D(3) * \Delta P_{1,2}(2,3) \\ &= 2\frac{2}{6} * (-0,5) + \frac{5}{6} * 0.5 \\ &= -\frac{14}{12} + \frac{5}{12} \\ &= -\frac{9}{12} = -\frac{3}{4} \end{aligned}$$

Overall forces

The total force is the sum of direct and indirect forces:

$$F(j,i) = SF(j,i) + VF(j,i) + NF(j,i)$$

In the example:

$$F(1,2) = SF(1,2) + NF(1,2) = -\frac{1}{4} + (-\frac{3}{4}) = -1$$

The low value suggests mapping task 1 to time step 2

Overall approach

```
procedure forceDirectedScheduling;
  begin
     AsapScheduling;
     AlapScheduling;
     while not all tasks scheduled do
        begin
           select task T with smallest total force;
           schedule task T at time step minimizing forces;
           recompute forces;
        end;
  end
```

May be repeated for different task/ processor classes

Not sufficient for today's complex, heterogeneous hardware platforms

Evaluation of HLS-Scheduling

- Focus on considering dependencies
- Mostly heuristics, few proofs on optimality
- Not using global knowledge about periods etc.
- Considering discrete time intervals
- Variable execution time available only as an extension
- Includes modeling of heterogeneous systems

Overview

Scheduling of aperiodic tasks with real-time constraints:
Table with some known algorithms:

	Equal arrival times; non-preemptive	Arbitrary arrival times; preemptive
Independent tasks	EDD (Jackson)	EDF (Horn)
Dependent tasks	LDF (Lawler), ASAP, ALAP, LS, FDS	EDF* (Chetto)

Conclusion

- HLS-based scheduling
  - ASAP
  - ALAP
  - *List scheduling* (LS)
  - *Force-directed scheduling* (FDS)
- Evaluation