Learning objective: study classical heuristics and anomalies for the scheduling problem with task precedence $P|\text{pred}|C_{\text{max}}$.

1 Task graph scheduling heuristics

Consider the following DAG where a pair $X/w$ means that task $X$ has weight $w$. For instance, A has weight 3.

Exercise 1: Scheduling without communications

We first disregard the labelling of the edges and assume communications come for free.

Compute the bottom level for each node.

Schedule the task graph on 3 processors using a list heuristic. What is the makespan of our schedule? Is it optimal?

Exercise 2: Critical path scheduling

From now on, we will consider the communication costs which have to be accounted for when two adjacent tasks are scheduled on different processors.

How communications should be taken into consideration when computing the bottom level?

Compute the bottom level for each node.

Schedule the task graph on 3 processors using the critical path heuristic. What is the makespan of our schedule?

Exercise 3: Modified critical path scheduling

Sometimes, it is worth waiting to schedule a task on a busy processor rather than using the first processor available.

Which wrong decision, made in the previous section, would be avoided by using this new heuristic?
Using this approach, propose a new schedule for our task graph with 3 processors. What is its makespan?

2 List scheduling anomalies

Consider the following DAG with two components.

Exercise 4: No anomalies
What is the makespan achieved by the critical path list scheduling with 2 processors? Is it optimal?

Exercise 5: Anomalies on weights
Assume that each task weight is decreased by one unit (now A has weight 7, B has weight 1, and so on). Show that the makespan achieved by the critical path list scheduling increases. Show that, somewhat shockingly, it is impossible to get a lower makespan than before with a list scheduling algorithm.

Exercise 6: Anomalies on processors
Going back to original task weights, assume that we have 3 processors. Show that the makespan achieved by the critical path list scheduling increases. Show that the makespan achieved by any list scheduling algorithm, shockingly again, increases.