Real Time Systems 2

Myopic Offline Scheduling Algorithm

Lecture 7

Myopic Offline Scheduling algorithm

- Myopic Offline Scheduling (MOS) heuristic is an assignment/ scheduling algorithm meant for non-preemptive tasks.

- This algorithm takes account not only of processing needs but also of any requirements that takes may have for additional resources.

- For example, a task may need to have exclusive access to a block of memory or may need to have control over a printer.
- MOS is an offline algorithm in that we are given in advance the entire set of tasks, their arrival time, execution time and deadline

Myopic Offline Scheduling algorithm

- MOS proceeds by building up a schedule tree.

- Each node in this tree represents an assignment and scheduling of a subset of tasks.

- The root of the schedule tree is an empty schedule.
- Each child of a node consists of the schedule of its parents node, extended by one task.

- A leaf of this tree consists of a schedule (feasible or infeasible) of the entire task set.

Myopic Offline Scheduling algorithm

- The schedule tree for an nT task system consists of nT+1 levels (including the root).
- Level i of the tree (counting the root as being of level 0) consists of nodes representing schedules including exactly i of tasks.

- Generating the complete tree is tantamount to an exclusive enumeration of all possible allocations.
- For any but the smallest systems, it is therefore not practical to generate the complete tree.

- Instead, we try to get to a feasible schedule as quickly as we can.

Myopic Offline Scheduling algorithm

- The algorithm:
- We start at the root node, which is an empty schedule, that is, it corresponds to no task having been scheduled.
- We then proceeds to build the tree from that point by developing nodes.
- A node is developed as follows:
- Given a node n, we try to extend the schedule represented by that node by one or more tasks.
- That is, we pick up one of the as-yet-unscheduled tasks and try to add it to the schedule represented by node n . The augmented schedule is a child node of n.

Myopic Offline Scheduling algorithm

- There are two questions that must be answered:
- First: Which task do we pick for extending an incomplete schedule?
- Second: When do we decide that a node is not worth developing further and turn to another node?

Myopic Offline Scheduling algorithm

- 1. The task that we chose to extend an incomplete schedule is one that minimize a heuristic function H. H may be any of the following functions:
- Task execution time.
- Deadline
- Earliest start time.
- Laxity. (Di – ei).
- Weighted sum of any of the above.

Myopic Offline Scheduling algorithm

- 2. We only develop a node if it is strongly feasible.
- A node is strongly feasible if a feasible schedule can be generated by extending the current partial schedule with any one of the as-yet-unscheduled tasks.

- If a node is not strongly feasible, it means that none of its descendants that are leaves can represent a feasible schedule.

- If we encounter a node that is not strongly feasible, we backtrack.
- That is, we mark that node as hopeless, and then go back to its parent, resuming the schedule-building from that point.

Myopic Offline Scheduling algorithm

- One difficulty with the MOS algorithm is that, if the number of tasks is very large, it can take a long time to check if a node is strongly feasible.

- In particular, at level I, we will need to check feasibility of extending the schedule by each of the nT – i as-yet-unscheduled tasks.

- As a result, the number of comparisons needed to generate one root-to-leaf path is:
- $nT + (nT-1) + (nT-2) + \qquad + 0 = (nT^{*} (nT + 1))/2$

Myopic Offline Scheduling algorithm

- To reduce the number of comparisons, we can replace the strongly feasibility check at each node by means of myopic procedure as follows:
- For each nonleaf level-i node n, this procedure picks the first
- min {k, nT-i) as-yet-unscheduled tasks and checks to see if the schedule represented by n can be feasibly extended by each of these tasks.
- If not, we mark the node as hopeless and backtrack as before.
- Otherwise, we develop children for that node.

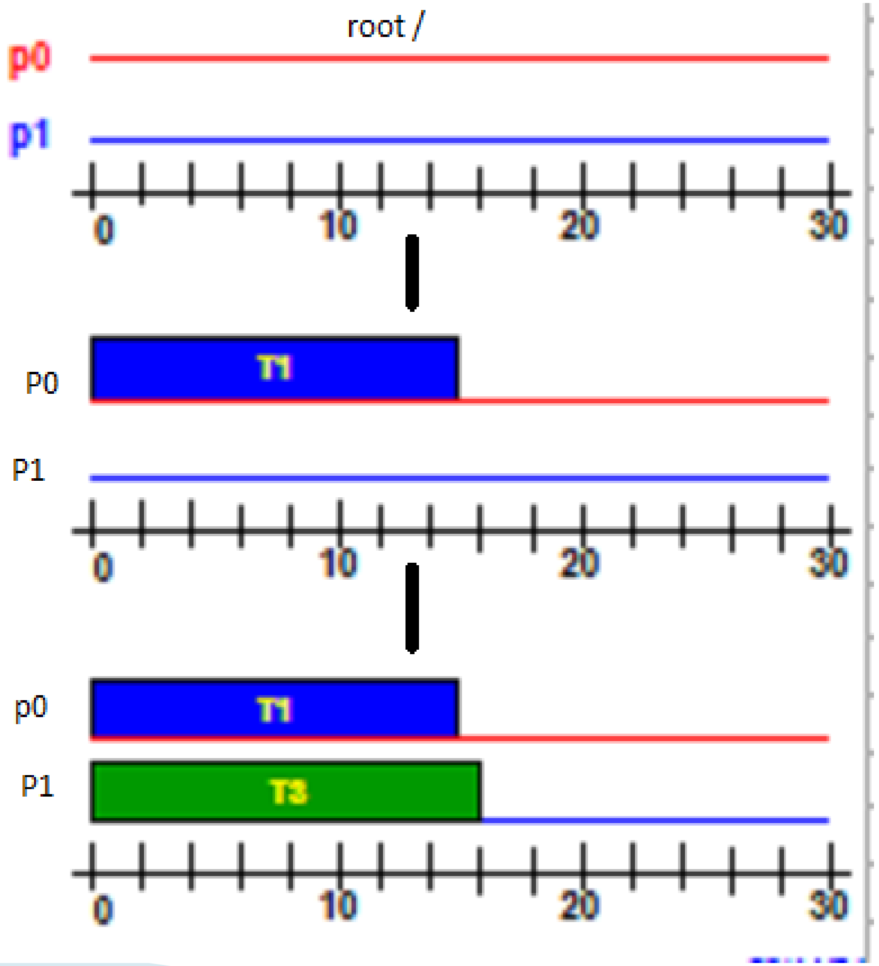
Myopic Offline Scheduling algorithm

- Example: We have a five task set to be scheduled on two-processor system.
- The tasks are non-preemptive. The parameters of these tasks are as follows: There are no other resource requirements. Suppose we use H(i)=ri . We set k=5 for the myopic procedure.

	T1	T2	T3	T4	T5
Ri	0	10	0	15	0
Ei	15	5	16	9	10
Di	15	20	18	25	50

Myopic Offline Scheduling algorithm

- The root node is the empty schedule. There are three tasks with release time = 0. we pick T1 first. A level-1 node is generated.



Myopic Offline Scheduling algorithm/ Example

- That contains a schedule for T1. This node is strongly feasible – any of the other tasks can be feasibly scheduled given the position that T1 occupies in this schedule.

- Next, we pick T3 and schedule it to form a level – 2 mode. This is also strongly feasible. Then we generate a level-3 schedule, which involves augmenting the previous schedule with T5.

- Unfortunately, this is not strongly feasible.
- In particular, it would be impossible to augment this schedule with T2.

Myopic Offline Scheduling algorithm/ Example

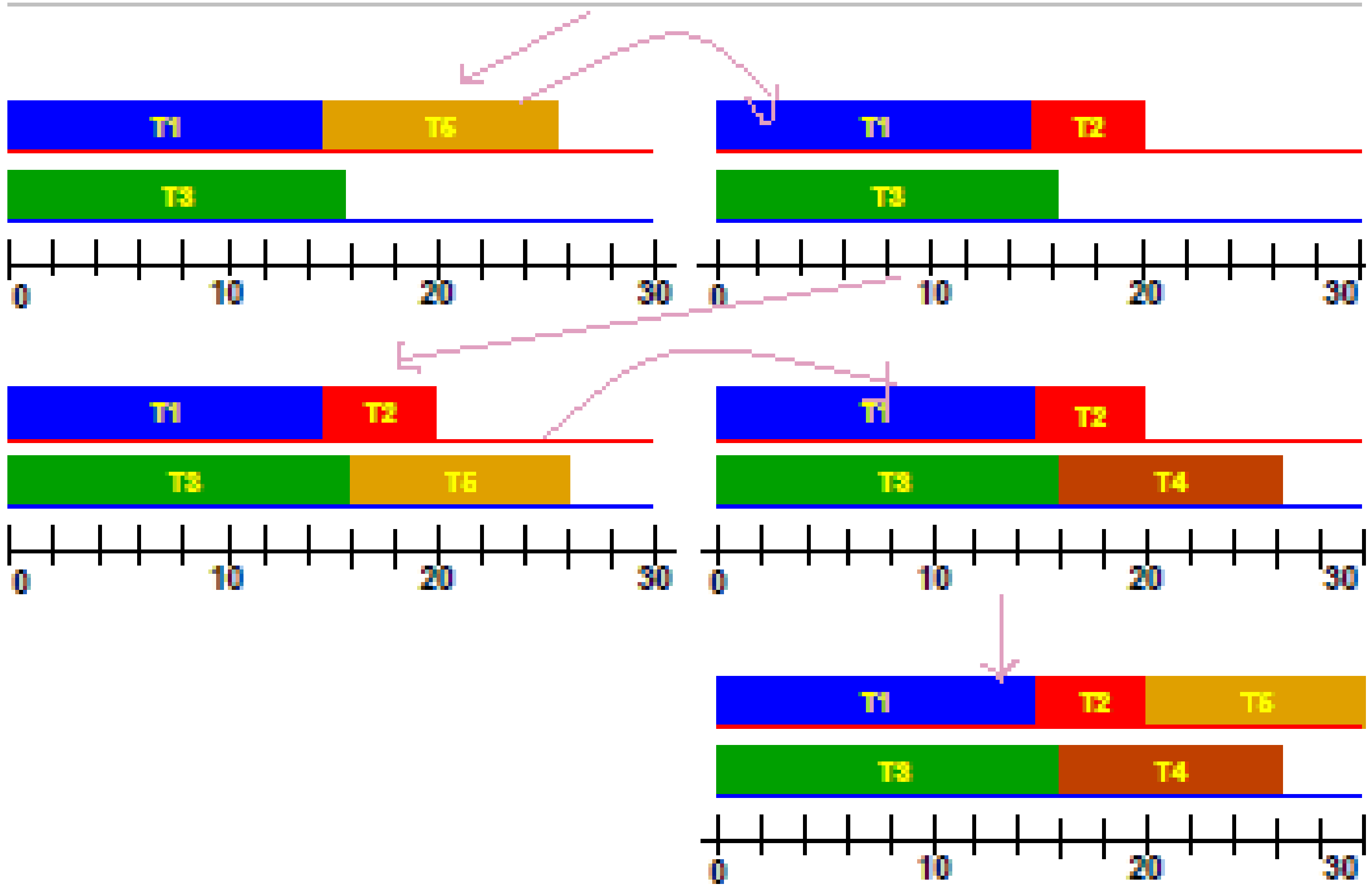
- So, we backtrack to the level-2 (i.e. the parent node). We pick T2 rather than T5 (the next task in order of release time) and schedule it.

- This results in a strongly feasible schedule.

- Next, we form a level-5 node by adding T5 to the schedule.

- This is not strongly feasible – T4 can not be added to it.
- So, we abandon this node, return to the parent (level-4) node.
- Generate a schedule by adding T4. This is strongly feasible.
- Then adding T5.

Myopic Offline Scheduling algorithm



Myopic Offline Scheduling algorithm

- The running time of the algorithm depends on K and H.
- K bounds the number of tasks that the algorithm considers in determining the strong feasibility of a node.

- If K is too small, it is possible for us to declare a node to be strongly feasible and develop it further, only to find that none of its descendants is strongly feasible.

Myopic Offline Scheduling algorithm

- If K is too large, we will spend a great deal of time (especially in the levels of the tree close to the root) checking the strong feasibility of nodes.

- In general, the tighter the constrains, the greater must be the value of K.

- In other words, if the task laxities are low or if many tasks use resources in addition to the processor, k must be large.
- It has been suggested, K = 13 is the largest value ever required.

Myopic Offline Scheduling algorithm

- H is a weighted sum of the deadline and earliest start time is perhaps the most promising function.

- Homework: Resolve the example with $H(i) = Di$ and see if it runs faster for that function.