Real Time systems 2

Scheduling of IRIS Tasks

Increased Reward with Increased Service

Scheduling of IRIS Tasks

- To obtain acceptable output: a task had to be run to completion.
- So if the task is not run to completion, we get zero reward from it (it may as well not have been run).
- However, there is a large number of tasks this, is not true.
- These are *iterative* algorithms. The longer they run, the higher is the quality of their output (up to some maximum runtime).

Scheduling of IRIS Tasks

- Search algorithms for finding the minimum of some complicated function are also another example of *iterative* tasks. The longer we search the parameter space, the greater is the chance that we will obtain the optimum value, or something close to it.
- Example: Chess-playing algorithms evaluate the goodness of moves by looking ahead several moves. The more time they have, the further they can look, and the more accurate will be the evaluation.

Scheduling of IRIS Tasks

- Such tasks known as Increased Reward with Increased Service (IRIS) tasks.

The reward function associated with an IRIS task increases with the amount of service given to it.

- Tasks with this reward function having a ***mandatory*** and an ***optional*** component.
- The mandatory portion must be completed by the deadline if the task is critical;
- The optional portion can be done if time permits. The optional portion requires a total of ***o*** time to complete. In each case, the execution of a task must be stopped by its deadline *d.*

Scheduling of IRIS Tasks

- The scheduling task can be described as the following optimization problem:
- ***Schedule the tasks so that the reward is maximized, subject to the requirement that the mandatory portions of all the tasks are completed.***
- This optimization problem is NP-complete when there is no restriction on the release times, deadlines, and reward functions.
- However in what follows, mi and oi denote the execution time of the mandatory and optional parts, respectively of Ti.

Identical Linear Reward Functions

- A schedule is said to be optimal if the reward is maximized subject to all tasks completing at least their mandatory portions by the task deadline.

- Theorem: *The EDF algorithm is optimal if the mandatory parts of all tasks are completed.*

- We can use this result to obtain an optimal scheduling algorithm for the case when the optional portions are not all zero.

Identical Linear Reward Functions

- The optimal algorithm IRIS its basic idea:
- First, since we receive one unit of reward for each unit of the optional portion completed (for any task), the highest reward subject to the constraint that all mandatory portions are completed, is obtained when the processor carries out as much execution as possible.

- The tasks $T1 \ldots T_n$ have mandatory portions M1 … M_n and optional portions $O1 \ldots O_n$.
- M = $\{M_1 \text{---} M_n\}$ O = $\{O_1 \text{---} O_n\}$ T = $\{T_1 \text{---} T_n\}$

IRIS Algorithm

- ***Step 1:*** Run the EDF algorithm on the task set T to generate a schedule, S_t. If this is feasible, An optimal schedule has been found STOP.
- Else,
- go to step 2.
- end if
- ***Step 2:*** Run the EDF algorithm on the task set M, to generate a schedule S_m. If this set is not feasible, M cannot be feasibly scheduled: STOP.
- Else,
- Define ai as the ith instant in S_m when either the scheduled task changes, or the processor becomes idle, i = 1,2,---.
- Let *k* be the total number of these instants.
- Define a0 as when the first task begins executing in S_m.
- Define r(j) as the task that executes in S_m in *[aj, aj+*1]

Algorithm IRIS

- Define $L_t(j)$ and $L_m(j)$ as the total execution time given to task r(j) in $St(j)$ and $S_m(j)$ respectively after time aj.
- Go to step 3.
- end if
- ***Step 3:*** j = k–*1*
- do while ($0 \leq j \leq k-1$)
- if $(L_m(j) > L_t(j))$ then
  Modify S_t by
- assigning $L_m(j) - L_t(j)$ of processor time in $[aj, aj_+1]$ to r(j), and
- reducing the processor time assigned to other tasks in $[aj, a_{j+}1]$ by $L_m(j) - L_t(j)$.
- Update $L_t(1), \ldots, L_t(j)$ appropriately, end if
- $J = j - 1$
- end do end

Identical Linear Reward Functions

- Running the EDF algorithm for the total execution time of each task.
- Resulting schedule *St, which is* maximizes the total processor busy time.

- If *St* is a feasible schedule, so we have given each task as much time as it needs to finish executing both its mandatory and optional portions and still met each task deadline.

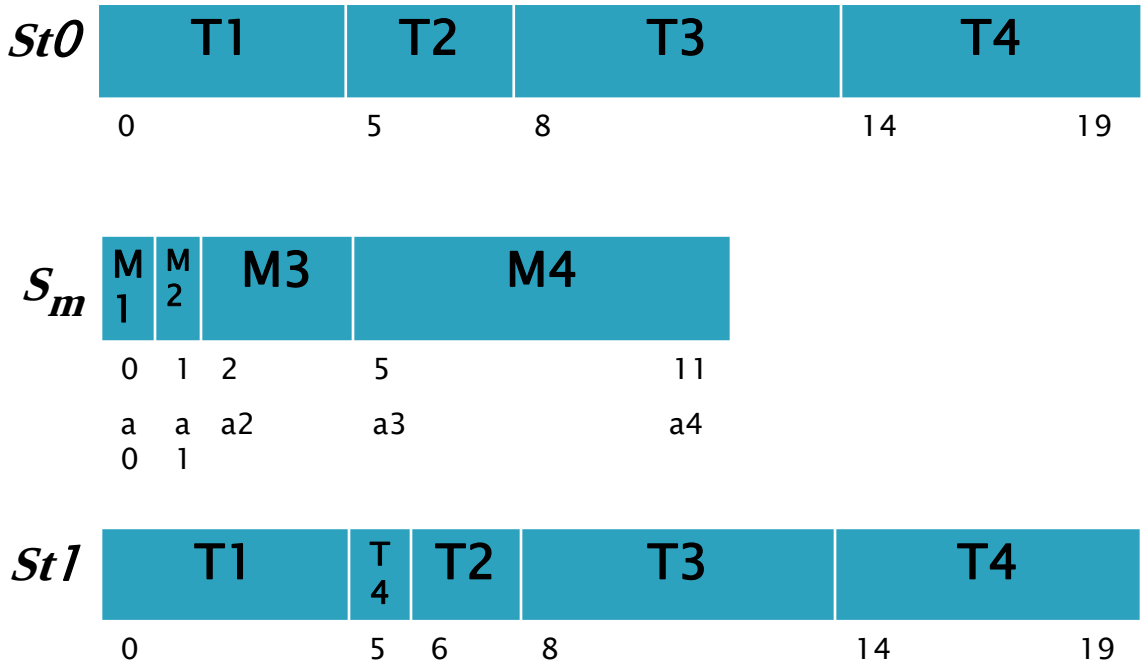
- If we do not obtain a feasible schedule, so some tasks cannot be given its full execution time and still met its deadline.

Identical Linear Reward Functions

- In that case, we run the EDF algorithm on the mandatory portions of each task, to yield schedule S_m.
- If this results is infeasible schedule, then we must stop since we can't even execute the mandatory portions of each task.
- If S_m is feasible, then we adjust St to ensure that each task receives at least its mandatory portion of service.

Example: Set of 4 tasks with parameters shown in the table.

Task	Ri	exei	mi	oi	Di
1	0	5	1	4	10
2	1	3	1	2	12
3	1	6	3	3	15
4	2	8	6	2	19



Schedules produced by IRIS

Identical Linear Reward Functions: Example

- ***Step 1:*** Running the EDF algorithm to produce *St0*

  Since T4 didn't meet its deadline *(it need 3 time units to complete its execution but its deadline arrives)*, then go to step 2.

- ***Step 2:*** Running the EDF algorithm on the tasks set depending on **mi,** produces the feasible schedule S_m

  All deadlines are met, so we can proceed to step 3, we have a0 = 0 a1 = 1 a2 =2 a3 = 5, a4 = 11. Also, k = 4 (No. of tasks).

- ***Step 3:*** So starting with a3(*k=4 , j=k-1*).

  1- T4 scheduled in S_m over the interval [a3, a4] and given 6 units of time.

Identical Linear Reward Functions: Example

- 2- In *St0*, T4 is given only 5 units of time, so we modify *St0* by adding 6-5 = 1 time unit to T4 in the interval [a3, a4), then subtract 1 unit from the task originally scheduled at a3 in *St0*, namely T2.
- The resultant schedule is *St1,* T4 being scheduled for a total of 6 units beyond a3.
- 3- Now move to the interval [a2 ,a3), *(back to a0).*
- T3 is scheduled beyond that time in S_m for a total of 3 units.
- In *St1*, T3 has been scheduled for a total of 6 units beyond a2, so no modifications are needed, T3 has enough time to meet its mandatory portion.

Identical Linear Reward Functions: Example

- 4- Then move to [a1, a2) T2 is scheduled for that interval in S_m .
- The time given to T2 beyond a1 in *St1* is 2 units, which is greater than the mandatory requirement, so no modifications are needed.
- 5- Finally the interval [a0 ,a1), T1 is given 1 unit in schedule S_m .
- In *St1*, T1 is scheduled for 5 units, which exceeds the time given in S_m .
- So no modifications are needed and the optimal schedule is *St1*.

Identical Linear Reward Functions: Example

HW:

Q2.(10 marks) Consider the following set of tasks:

Task no	Mi	Oi	total	Ri	Di
T1	5	5	10	0	12
T2	4	3	7	1	13
T3	3	1	4	17	23
T4	5	2	7	18	25

A(2 marks) Run EDF algorithm to generate St0. Is it feasible?

B(2 marks) Run EDF on mandatory execution only. Is it feasible?Mark the busy period.

C (8 marks) Generate the final schedule using IRIS algorithm.