Embedded System Reference Paper Analysis

Ragesh RAMACHANDRAN

December 17, 2018

An analysis of two research papers "*Skip-Over: Algorithms and Complexity for Overloaded Systems that Allow Skips*" and "*A Scheduling Model for Reduced CPU Energy*" were carried out as a part of course work and result of the analysis is explained in section 1 and section 2 respectively.

1 Skip-Over:Algorithms and Complexity for Overloaded Systems that Allow Skips

Job skipping is the occasional skipping of jobs in a real time system when the system is overloaded. Skipping the deadlines are accepted provided most deadlines are met. We look at the problem of uniprocessor scheduling of occasionally skip able periodic tasks in a n environment having periodic tasks.

1.1 Assumptions of skip over model

We consider uniprocessor system which is preemptive with no overheads. Each task has computation time C, Period P and deadline is equal to the period of the task. Tasks are assumed to be independent and tasks are divide into instances.

Red task- Task instance must complete before its deadline
Blue task- Task instance can be aborted at any time.

When a task misses its deadline then we can say that the task is skipped. The number of possible skips s is given by $2 \leq s \leq \infty$ which implied that the distance between two skips must be at least s periods.

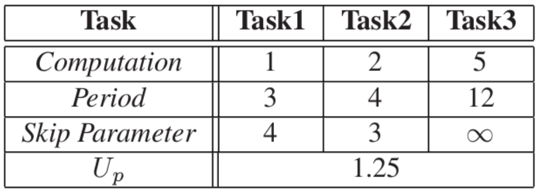
1.2 Necessary feasibility condition

Given a task set with $T_i(p_i, c_i, s_i)$ of periodic tasks that allow skips then the given equation is the necessary condition for feasibility.

$$\sum_{i=1}^{n} \frac{c_i(s_i - 1)}{p_i s_i} \leq 1 \tag{1}$$

1.3 Illustration of skip over scheduling approaches

In the task sets given in 1 note that the processor utilization factor is greater than 1 but the necessary feasibility condition is satisfied. The skip factor of $task1$is 4 which means every fourth instance is skipped, the $task2$ is skipped every third instance but $task3$ has a skip factor of ∞ which means it can not be skipped. The total processor utilization is found to be 1.25



Task	Task1	Task2	Task3
Computation	1	2	5
Period	3	4	12
Skip Parameter	4	3	∞
U_p	1.25		

Figure 1: Task set for Skip over

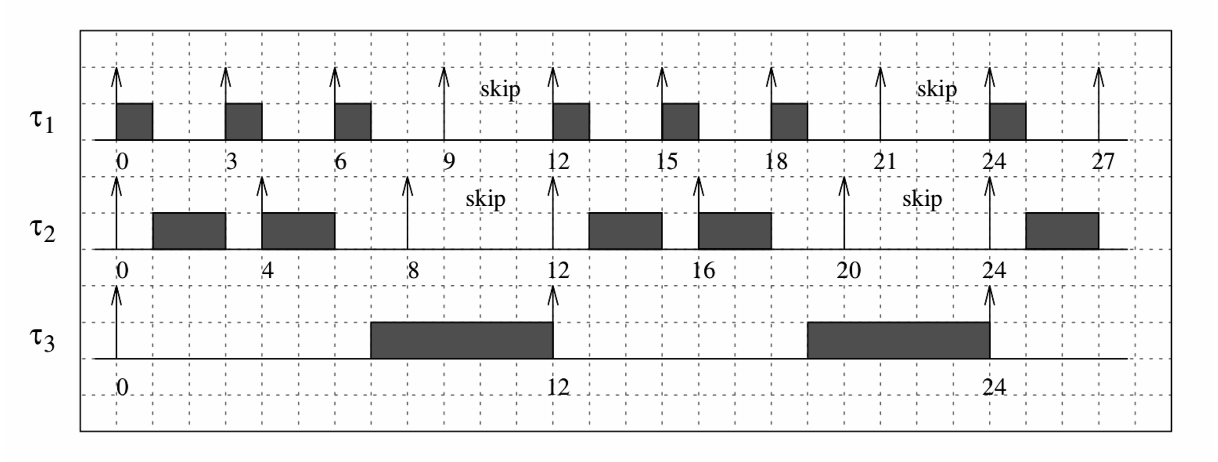


Figure 2: Tasks scheduled using- skip over algorithm

2 A scheduling model for reduced CPU energy - YDS Algorithm

A simple model of job scheduling is proposed to minimize energy consumption. In this model, each job is to be executed between its arrival time and deadline by a single processor with variable speed, under the assumption that energy usage per unit time, P , is a convex function of the processor speed s.

2.1 Problem

The problem we are going to address here is to find a feasible scheduling algorithm that can minimize the Energy usage of CPU.

2.2 Assumption

- Let $[t_0, t_1]$ be the fixed time interval and J is the set of jobs to be executed during the fixed interval. Each jobs j is an element of J and has the parameters:
  - Arrival time a_j
  - Deadline b_j
  - Required number of CPU cycles R_j
  - $[a_j, b_j]$ is the interval of job j
  - $s(t) \geq 0$ is the processor speed at time t
  - *job(t)* defines the job being executed at time t
  - A schedule is a pair $S = (s(t), job(t))$
- A feasible schedule for an instance is given by

$$\int_{\mathbf{a_j}}^{\mathbf{b_j}} s(t)\delta(job(t), j)dt = R_j$$

  where $\delta(x, y)$ is 1 if x = y and 0 otherwise
- The energy consumed by a schedule S is given by

$$E(s) = \int_{\mathbf{t_0}}^{\mathbf{t_1}} P(s(t))dt$$

- The goal of the scheduling problem is to find a feasible schedule that minimize *E(s)*
- The intensity $g(I)$ of an interval $I = [z, z']$

$$g(I) = \sum \frac{R_j}{(z' - z)}$$

- The interval $[z, z']$ is called as critical interval if it maximize $g(I)$

2.3 Description of YDS algorithm

Off-line algorithm that computes the minimum energy schedule for any set of jobs

- Schedule the jobs in J_{I*} by Earliest deadline policy.
- Run all jobs in J_{I*} at speed $g(I*)$

- Modify the problem to reflect the deletion of J_{I*} and $I*$.
  - Remove jobs J_{I*} from J.
  - Update arrival times and deadlines to ensure no job outside of J_{I*} is scheduled in the interval $I*$.

2.4 Illustrative example

The following task sets of 10 jobs with arrival time, deadline and required number of CPU cycles are used for the YDS algorithm.

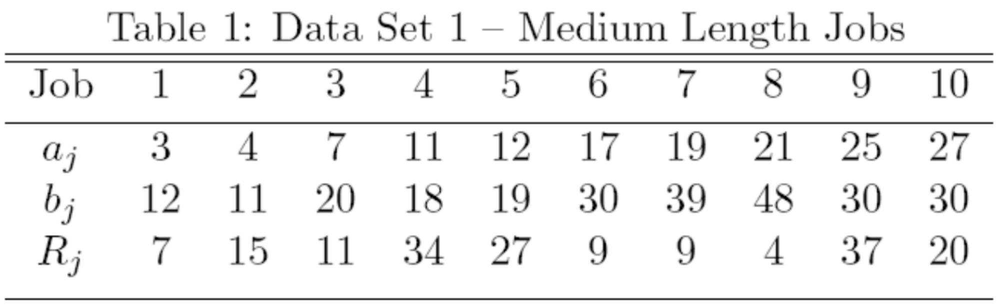


Table 1: Data Set 1 – Medium Length Jobs

Job	1	2	3	4	5	6	7	8	9	10
a_j	3	4	7	11	12	17	19	21	25	27
b_j	12	11	20	18	19	30	39	48	30	30
R_j	7	15	11	34	27	9	9	4	37	20

Figure 3: Task sets for YDS algorithm

- The jobs 1,2 and 3 are executed at same speed of 3.667
- The jobs 4 and 5 are executed at 7.625
- Job 3 is preempted at $t = 11$ and continues execution at $t = 19$
- job 6 has speed 1.8
- Instead of job 7 and 8, job 9 and 10 begins execution at $t = 25$ at same speed 11.4.
- At $t = 30$ job 7 executes at speed 1 and then job 8 executes.

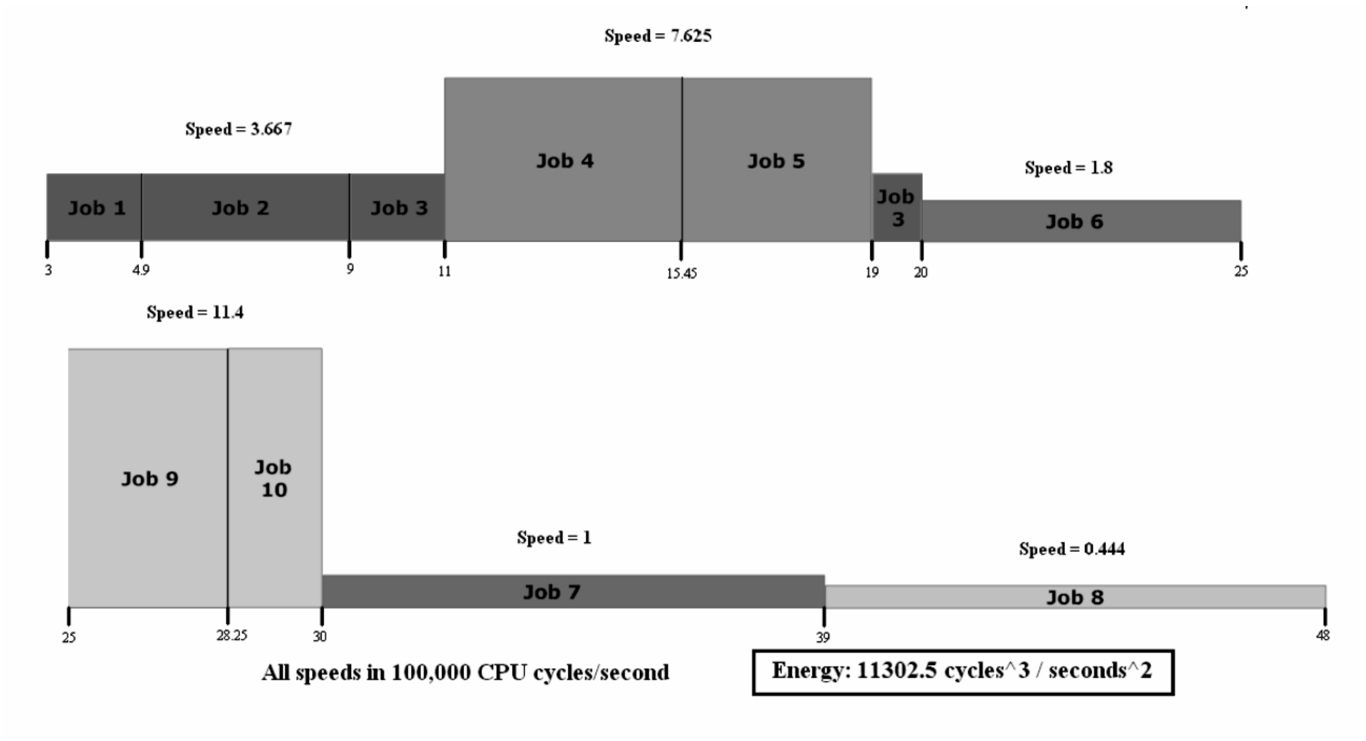


Figure 4: Schedule produced by YDS algorithm