Schedulability Analysis for Dual Priority Scheduling

I. Model

In this paper, we will use the classic sporadic task model [**?**], where a task can be specified as a 4-parameter tuple: $\tau_i = (C_i, D_i, T_i)$, where C_i denotes the worst-case execution time estimate (WCET), D_i denotes the relative deadline and T_i denotes the minimal time separation between two successive job release (or called period). Each job should execute up to C_i time units before the deadline, and otherwise the system is regarded as failure.

Each task has an origin priority and a promoted priority. After τ_i release a new job J_i, it is scheduled according to its origin priority. However J_i will be scheduled according to its promoted priority after p_i time units from its release time.

Given a task system $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ sorted according to their origin priority in descending order, we first make the following **assumptions**:

1) Each task τ_i has a original priority $n+i$ and a promotion priority i (a smaller index implies higher priority).
2) Each task has a fixed promotion point p_i. The concerned job $J_{i,x}$ has its promotion point $P_{i,x} = r_{i,x} + p_i$, where $r_{i,x}$ denotes the release time of $J_{i,x}$.

In this paper, our schedulability analysis of dual priority scheduling is restricted to the case that satisfies the above assumptions. Note that, our analysis in fact can be easily extended to more general systems, and we may explore in the future. For simplicity, we will only present the restricted version in this paper.

II. Dual Priority Schedulability

The following theorem is an exact test to determine whether τ_i is schedulable on a single processor using dual priorities.

Theorem 1: A task τ_i is schedulable on a single processor using dual priority scheduling if for each absolute deadline of a job $d_{i,x}$ where $x \in N$, there exists a t' where $t - D_i \leq t' \leq t\ (= d_{i,x})$ for which the following condition holds:

$$C_i + F_i(\tau, t', t) \leq t' \tag{1}$$

where $F_i(\tau, t')$ denotes the maximum possible execution resource consumed by the other jobs except $J_{i,x}$ released by tasks in the system during $[0, t']$

Proof 1: We can prove the statement that if τ_i is not schedulable, then Equation **??** would not hold. Suppose that τ_i is not schedulable, then there are legal event sequences in which deadline $d_{i,x}$ is missed when τ_i is assigned with the current priority and promotion point. Let S' denote such a sequence where $J_{i,x}$ misses deadline at the earliest time at t (i.e., $d_{i,x} = t$).

It must be the cases that during $[0, t]$ some other jobs are executing because otherwise the time interval between the last idle instant to t could also construct such a sequence. Then it must be that during the time interval $[0, t']$, other jobs have consumed an amount of resource more than

$$F_i(\tau, t', t) > t' - C_i$$

which contradicts the Equation **??**.

Therefore a system is schedulable on a single processor using dual priority scheduling if and only if all the tasks in the system meet the requirement in Theorem **??**, and hence we can derive the following theorem.

Theorem 2: A system τ is schedulable on a single processor using dual priority scheduling if the following condition holds: $\forall\ \tau_i \in \tau :\ \ \forall t \geq D_i :\ \exists t' \in [t - D_i, t]$ so that

$$C_i + F_i(\tau, t', t) \leq t'$$

Unfortunately, it is very hard to know the exact value of $F_i(\tau, t', t)$, and hence we choose to derive an upper bound by considering each task separately. Let $f_i(\tau_j, t', t)$ denotes the maximum possible resource consumed by τ_j during $[0, t']$ in the scenario

when all other tasks do not release any jobs (except $J_{i,x}$). In this case, the execution τ_j is independent of interference from other tasks, and hence

$$F_i(\tau, t') \leq \sum_{\tau_j \in \tau \setminus \tau_i} f_i(\tau_j, t', t) + \lfloor \frac{t - D_i}{T_i} \rfloor \times C_i$$

where $\lfloor \frac{t-D_i}{T_i} \rfloor \times C_i$ upper bounds the execution of τ_i itself before $J_{i,x}$. .

III. Interference Bound Function

Here we first introduce some notations that will be used later in the paper.

- $P_{i,x}$ denotes the promotion point of $J_{i,x}$
- $J_{j,y}$ is the job that has its release time $r_{i,y} \leq t' < r_{i,y} + T_j$.
- $J_{j,y'}$ is the job that has its release time $r_{i,y'} \leq P_{i,x} < r_{i,y'} + T_j$.
- $[a]_0 = \max(a, 0)$

When calculating $f_i(\tau_j, t', t)$, we ignore the interference from the other tasks on τ_j except $J_{i,x}$. Thus we have the following lemma.

Lemma 1 (Maximum Execution): $f_i(\tau_j, t', t)$ is maximized when $J_{j,1}$ is released at 0 and all jobs are released as soon as possible with period T_j, and each job of τ_j executes as early as possible as long as it has higher priority than $J_{i,x}$.

Proof 2: When $J_{j,1}$ is released at 0 and all jobs are released as soon as possible with period T_j, the number of jobs that released during $[0, t']$ is maximized. Meanwhile as we either shift the release pattern left or right, the total execution would only decrease or stay the same.

With Lemma **??**, we know $r_{j,y} = \lfloor \frac{t'}{T_j} \rfloor \times T_j$, and hence we can easily calculate $f_i(\tau_j, t', t)$. In the following section, we present the detailed equations.

A. $f_i(\tau_j, t', t)$ when $j < i$

We first consider the case when index j is smaller than index i which means τ_j has higher origin priority than τ_i. Thus if $J_{j,y}$ would not execute during the interval $[P_{i,x}, P_{j,y}]$ (if $P_{i,x} < P_{j,y}$) before $J_{i,x}$ finishes.

Case 1 ($P_{j,y} \leq P_{i,x}$**)** as shown in Figure **??**: Job $J_{i,x}$ always has lower priority than $J_{j,y}$. Thus the maximum possible resource consumed by τ_j before t' is bounded by

$$f_i(\tau_j, t', t) = \lfloor \frac{t'}{T_j} \rfloor . C_j + \min\left(C_j, t' - r_{j,y}\right)$$

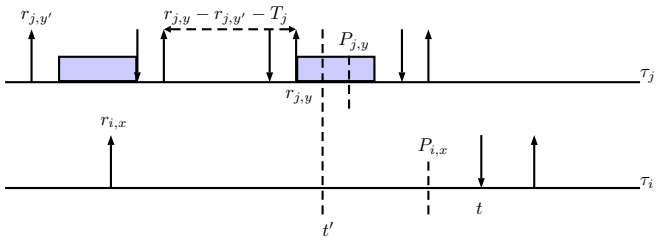

Fig. 1. $P_{j,y} \leq P_{i,x}$

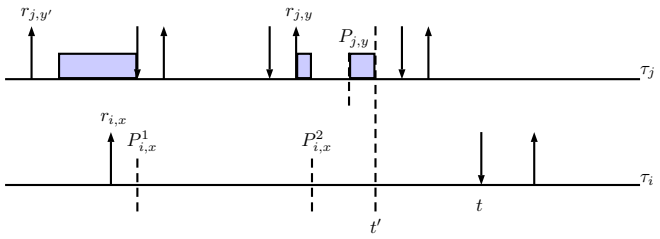

Fig. 2. $P_{j,y} > P_{i,x}$

Case 2 ($P_{j,y} > P_{i,x}$): as shown in Figure **??**, $J_{i,x}$ has higher priority than $J_{j,y}$'s during $[\max(r_{j,y}, P_{i,x}), P_{j,y}]$ (if $P_{i,x} \leq P_{j,y}$). Thus $J_{j,y}$ can not execute during $[\max(r_{j,y}, P_{i,x}), P_{j,y}]$ unless $J_{i,x}$ has already finished. Thus we have

$$f_i(\tau_j, t', t) = \lfloor \frac{t'}{T_j} \rfloor . C_j + \min\left(C_j, t' - r_{j,y} - [\min(t', P_{j,y}) - \max(r_{j,y}, P_{i,x})]_0\right)$$

B. $f_i(\tau_j, t', t)$ when $j > i$

In this section we consider the case when τ_j has lower origin priority (higher index) than τ_i.

Case 1.1 ($P_{i,x} \leq P_{j,y} \wedge r_{j,y} \leq r_{i,x}$): as shown in Figure **??**, τ_j may execute during $[r_{j,y}, r_{i,x}]$, but $J_{j,y}$ would not execute after $r_{i,x}$ unless $J_{i,x}$ has finished. Thus we have

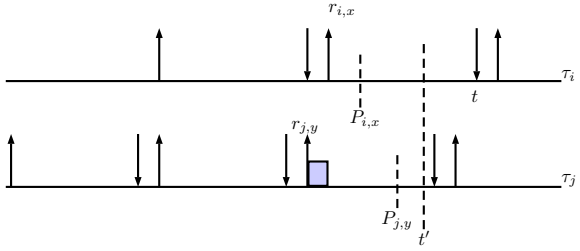

Fig. 3. $P_{i,x} \leq P_{j,t} \wedge r_{j,y} \leq r_{i,x}$

$$f_i(\tau_j, t', t) = \lfloor \frac{t'}{T_j} \rfloor \times C_j + \min(C_j, r_{i,x} - r_{j,y})$$

Case 1.2 ($P_{i,x} \leq P_{j,y} \wedge r_{j,y} > r_{i,x}$): as shown in Figure **??**, $J_{j,y}$ would not execute unless $J_{i,x}$ has completed. Thus we have

$$f_i(\tau_j, t', t) = \lfloor \frac{t'}{T_j} \rfloor \times C_j$$

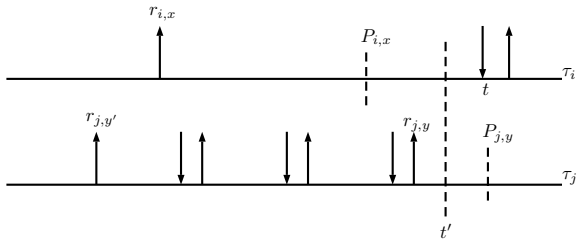

Fig. 4. $P_{i,x} \leq P_{j,y} \wedge r_{j,y} > r_{i,x}$

Case 2.1 ($P_{i,x} > P_{j,y} \wedge r_{j,y} \leq r_{i,x}$): as shown in Figure **??**, after $r_{i,x}$, $J_{j,y}$ can only execute during $[\max(r_{i,x}, P_{j,y}), \min(t', P_{i,x})]$. Thus we have

$$f_i(\tau_j, t', t) = \lfloor \frac{t'}{T_j} \rfloor \times C_j + \min\left(C_j, r_{i,x} - r_{j,y} + [\min(t', P_{i,x}) - \max(P_{j,y}, r_{i,x})]_0\right)$$

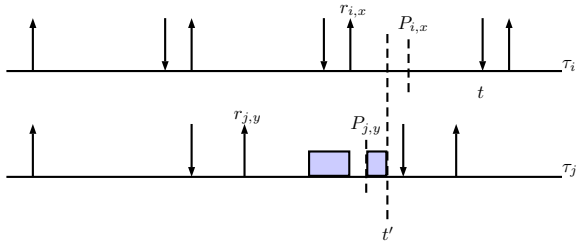

Fig. 5. $P_{i,x} > P_{j,y} \wedge r_{j,y} \leq r_{i,x}$

Case 2.2 ($P_{i,x} > P_{j,y} \wedge r_{j,y} > r_{i,x}$): as shown in Figure **??**, $J_{j,y}$ would only execute during $[P_{j,y}, P_{i,x}]$ before $J_{i,x}$ finishes. Thus we have

$$f_i(\tau_j, t', t) = \min\left(C_j, [\min(t', P_{i,x}) - P_{j,y}]_0\right) + \lfloor \frac{t'}{T_j} \rfloor \times C_j$$

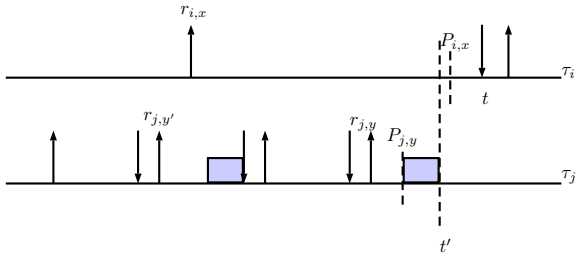

Fig. 6. $P_{i,x} > P_{j,y} \wedge r_{j,y} > r_{i,x}$

IV. Optimization Technique and Promotion Point Tuning

A. Optimization Technique

Since the total exectuion happens before $P_{i,x}$ could not exceed $P_{i,x}$, we will introduce an optimization technique based on this fact. Function $f_i(\tau_j, t', t)$ denotes the maximum resource used by τ_j during $[0, t']$, and here we denfine another function $g_i(\tau_j, t', t)$ which denotes the maximum possible resource used by τ_j during $[P_{i,x}, t']$ (only if $P_{i,x} < t'$). Note that if τ_j has its index $j > i$, then

$$g_i(\tau_j, t', t) = 0$$

because it would never execute after $P_{i,x}$ before $J_{i,x}$ finishes.

Lemma 2: If $D_j - p_j \geq C_j$, then $g_i(\tau_j, t', t)$ maximizes when $r_{j,y'} = [P_{i,x} + C_j - D_j]_0$ and all successive jobs are released as soon as possible, as shown in Figure **??**. In this case $r_{j,y} = r_{j,y'} + \lfloor \frac{t' - r_{j,y'}}{T_j} \rfloor T_j$.

Proof 3: Case 1: If $P_{i,x} + C_j - D_j \geq 0 \Rightarrow r_{j,y'} = P_{i,x} + C_j - D_j$, (scenario in Figure **??**) as we shift the pattern left, $J_{j,y'}$ execution after $P_{i,x}$ will decrease linearly up to C_j, while execution of $J_{j,y}$ will increase at most linearly. On the other hand, as we shift the pattern right, $g_i(\tau_j, t', t)$ would only decrease or stay the same.

Case 2 :If $P_{i,x} + C_j - D_j < 0 \Rightarrow r_{j,y'} = 0$, as we shift the pattern left, execution of $J_{j,y'}$ decreases from C_j to 0 (when $y \neq y'$), while the increase of $J_{j,y}$ is bounded by C_j. On the other hand, as we shift the pattern right, $g_i(\tau_j, t', t)$ would only decrease or stay the same.

With the above lemma, we have

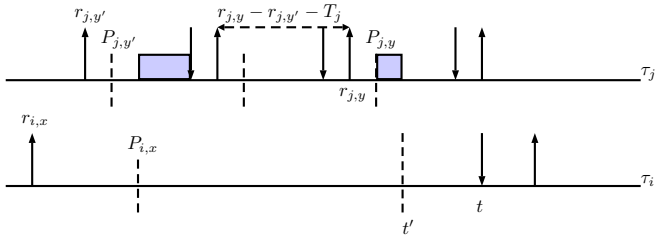

Fig. 7. $D_j - p_j \geq C_j$

$$g_i(\tau_j, t', t) = \begin{cases} \min(C_j, t' - P_{i,x}) & \text{if } y = y' \\ C_j + \frac{r_{j,y} - r_{j,y'} - T_j}{T_j} C_j & \\ + \min(C_j, [t' - P_{j,y}]_0) & \text{otherwise} \end{cases} \tag{2}$$

Lemma 3: If $D_j - p_j < C_j$, then $g_i(\tau_j, t', t)$ maximizes when $r_{j,y'} = [P_{i,x} - p_j]_0$ and all successive jobs are released as soon as possible, as shown in Figure **??**. In this case, $r_{j,y} = r_{j,y'} + \lfloor \frac{t' - r_{j,y'}}{T_j} \rfloor T_j$.

Proof 4: Each job of J_y could execute at most $D_j - p_j$ time units after $P_{i,x}$ before $J_{i,x}$ finishes. Case 1: If $P_{i,x} - p_j \geq 0 \Rightarrow r_{j,y'} = P_{i,x} - p_j$ (scenario in Figure **??**) as we shift the pattern left, $J_{j,y'}$ execution after $P_{i,x}$ will decrease linearly up to $D_j - p_j$, while execution of $J_{j,y}$ will increase at most linearly. On the other hand, as we shift the pattern right, $g_i(\tau_j, t', t)$ would only decrease or stay the same.

Case 2 :If $P_{i,x} - p_j < 0 \Rightarrow r_{j,y'} = 0$, as we shift the pattern left, execution of $J_{j,y'}$ decreases from $D_j - p_j$ to 0 (when $y \neq y'$), while the increase of $J_{j,y}$ is bounded by $D_j - p_j$. On the other hand, as we shift the pattern right, $g_i(\tau_j, t', t)$ would only decrease or stay the same.

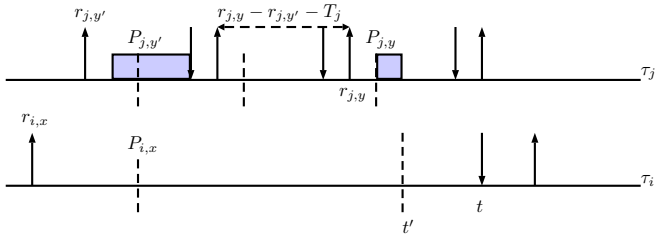

Fig. 8. $D_j - p_j < C_j$

With the above lemma, we have

$$g_i(\tau_j, t', t) = \begin{cases} \min(D_j - p_j, t' - P_{i,x}) & \text{if } y = y' \\ D_j - p_j + \frac{r_{j,y} - r_{j,y'} - T_j}{T_j}(D_j - p_j) & \\ + \min(D_j - p_j, [t' - P_{j,y}]_0) & \text{otherwise} \end{cases}$$

Since total execution before $P_{i,x}$ (if $t' > P_{i,x}$) is bounded by $P_{i,x}$, we can use a simple optimization technique to further tighten the test. Therefore we have

$$F_i(\tau, t', t) = \begin{cases} \sum_{\tau_j \in \{\tau \setminus \tau_i\}} f_i(\tau_j, t', t) + \lfloor \frac{t - D_i}{T_i} \rfloor C_i \text{ If } t' \leq P_{i,x} \\ \min\left(P_{i,x}, \lfloor \frac{t - D_i}{T_i} \rfloor C_i + \sum_{\tau_j \in \{\tau \setminus \tau_i\}} f_i(\tau_j, t', t) - g_i(\tau_j, t', t) \right) \\ + \sum_{\tau_j \in \{\tau \setminus \tau_i\}} g_i(\tau_j, t', t) \qquad \text{Otherwise} \end{cases} \tag{3}$$

B. Upper bound of t

We also need to derive an upper bound of t because otherwise t can tends to infinity. Suppose there exists a t so that

$$\forall\ t' \in (t - D_i, t]: \ F_i(\tau, t, t') + C_i > t' \Rightarrow \min_{t' \in (t - D_i, t]} \frac{F(\tau_i, t, t') + C_i}{t'} > 1$$

, and let

$$\begin{aligned} H_i(t) =& U \times t + \sum_{\tau_j \in \{\tau \setminus \tau_i\}} C_j + C_i \\ & \geq t \times u_i + \sum_{\tau_j \in \{\tau \setminus \tau_i\}} (\lfloor \frac{t'}{T_j} \rfloor + 1) C_j + C_i \\ & \geq F(\tau_i, t, t') + C_i \end{aligned}$$

Then it must be that

$$\frac{H_i(t)}{t - D_i} > 1 \Rightarrow t - D_i < U \times t + \sum_{\tau_j \in \{\tau \setminus \tau_i\}} C_j + C_i$$

$$\Rightarrow t < \frac{D_i + \sum_{\tau_j \in \tau \setminus \tau_i} C_j + C_i}{1 - U}$$

C. Promotion Point Tuning

The promotion point of each task has a great influence on the schedulability of the system. In this section we propose a promotion point tuning algorithm to find suitable promotion point for each task. The system τ will have an initial promotion point assignment based on the principle that: a task with lower origin priority has an shorter promotion point so that it can enter promotion stage earlier. Such a promotion point assignment principle enhances the chances that each task can complete by deadline.

Thus here we this the following initial promotion point policy.

$$p_i = T_i - C_i - \sum_{j<i} C_j \tag{4}$$

This assignment policy in fact is optimal when the system comprises of two tasks. Once our test detects τ_i has a deadline miss at t, then its promotion point decreases by dec_i until p_i decreases to 0. The value of dec_i controls the speed of the test because in the worst cases we may need to try all the possible combinations of p_i. Thus here we ensure dec_i fall in the range $[\delta_i^{min}, \delta_i^{max}]$ so that the value of dec_i is neither too large nor too small. On the other hand $J_{j,y}$ $(j > i)$ can only interfere $J_{i,x}$ only if $P_{j,y} < P_{i,x}$, and $J_{j,y}$ $(j < i)$ can not interfere $J_{i,x}$ during $[P_{i,x}, P_{j,y}]$. Therefore we can record

$$\delta_i = \min_{\tau_j \in \tau \setminus \tau_i \wedge P_{i,x} > P_{j,y}} P_{i,x} - P_{j,y}$$

, and once we detects that τ_i is not schedulable,

$$p_i = p_i - \min\left(\delta_i^{max} \max\left(\delta_i^{min}, \delta_i\right)\right)$$

In sum the detailed steps of the promotion point tuning steps are presented in the following algorithm.

Algorithm 1 Promotion Point Tuning

```
1:  function PP_T(τ)
2:      while flag == TRUE do
3:          flag← FALSE
4:          for τ_i ∈ τ do
5:              if TEST(τ_i, τ)==FALSE then
6:                  if p_i > dec_i then
7:                      p_i ← p_i − min(δ_i^max max(δ_i^min, δ_i))
8:                      flag← TRUE
9:                      Break
10:                 else
11:                     return FALSE
12:                 end if
13:             end if
14:         end for
15:     end while
16:     return TRUE
17: end function
18: function TEST(τ_i, τ)
19:     flag = TRUE
20:     for t ∈ [D_i, t^max] do
21:         for t' ∈ [t − D_i, t] do
22:             if F_i(τ, t', t) ≤ t' then
23:                 flag← TRUE
24:                 Break
25:             end if
26:         end for
27:         if flag==FALSE then return FALSE
28:         else
29:             flag← FALSE
30:         end if
31:     end for
32:     return TRUE
33: end function
```