امیررضا آذری

9910187

«به نام خدا»

تمرین اول سیستم‌های بی‌درنگ

اثبات بهینه بودن EDF:

بهینه بودن EDF یعنی اگر یک زمان‌بند feasible برای مجموعه وظایف T وجود داشته باشد، آن گاه EDF می‌تواند آن را پیدا کند. همچنین EDF می‌تواند maximum lateness را مینیموم کند. بنابراین، با توجه به اینکه اگر الگوریتمی بتواند این کار را انجام دهد، پس بهینه است، می‌توانیم این گزاره را اثبات کنیم و در نتیجه آن بهینه بودن EDF را نیز به اثبات برسانیم.

حال یک سری term را معرفی می‌کنیم:

σ را زمان‌بند تولید شده توسط یک الگوریتم دلخواه A باشد و σ_{EDF}، زمان‌بند تولید شده توسط EDF باشد.

با توجه به اینکه در EDF، preemption داریم پس یک وظیفه می‌تواند در بازه‌های زمانی جدا از هم یا هم عملیات اجرا شود. بنابراین بدون کاستن از کلیت مسئله، σ می‌تواند به تکه‌های زمانی یک واحد تقسیم کرد.

حال تعاریف زیر را مشخص می‌کنیم تا اثبات را طی کنیم:

* وظیفه اجرا شده در $[t, t+1)$ را با $\sigma(t)$ نمایش می‌دهیم.

* وظیفه آماده که در زمان t، نزدیک‌ترین ددلاین را دارد با $E(t)$ نشان می‌دهیم.

* $\tau_E(t)$ نیز برابر زمانی است که (بزرگتر مساوی زمان t است) برش بعدی وظیفه $E(t)$ در زمان‌بند فعلی، اجرای خودش را آغاز می‌کند.

حال اگر $\sigma_{EDF} \neq \sigma$ ، آنگاه در σ لحظه t ای وجود دارد که در آن $\sigma(t) \neq \mathcal{E}(t)$ و ایده به کار رفته در این اثبات این است که جابه‌جا کردن position ، $\sigma(t)$ و $\mathcal{E}(t)$ مقدار Maximum lateness را نمی‌تواند افزایش دهد.

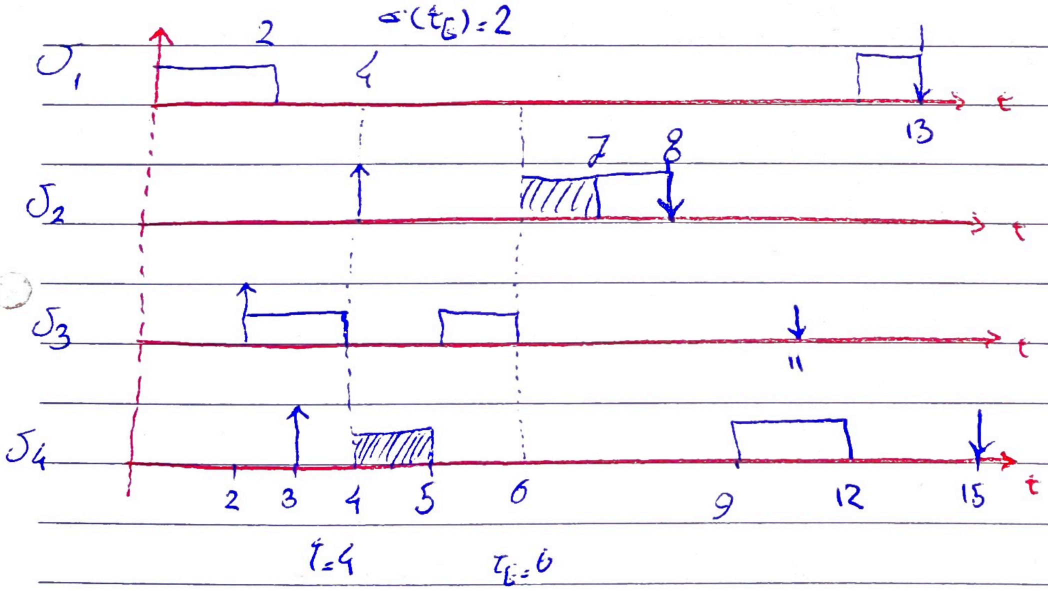
اگر زمان‌بند σ در لحظه $t=1$ شروع شود و t آخرین و دورترین در این مجموعه وظایف باشد، آنگاه σ_{EDF} می‌تواند از σ با حداکثر t تا transposition به دست آید.

بنابراین با توجه به اینکه گزاره بالا اثبات شد، یعنی اثبات شد که EDF می‌تواند Maximum lateness را minimum کند و می‌دانیم که اگر EDF چنین قابلیتی داشته باشد آنگاه optimal است، پس اثبات شد که EDF بهینه است.

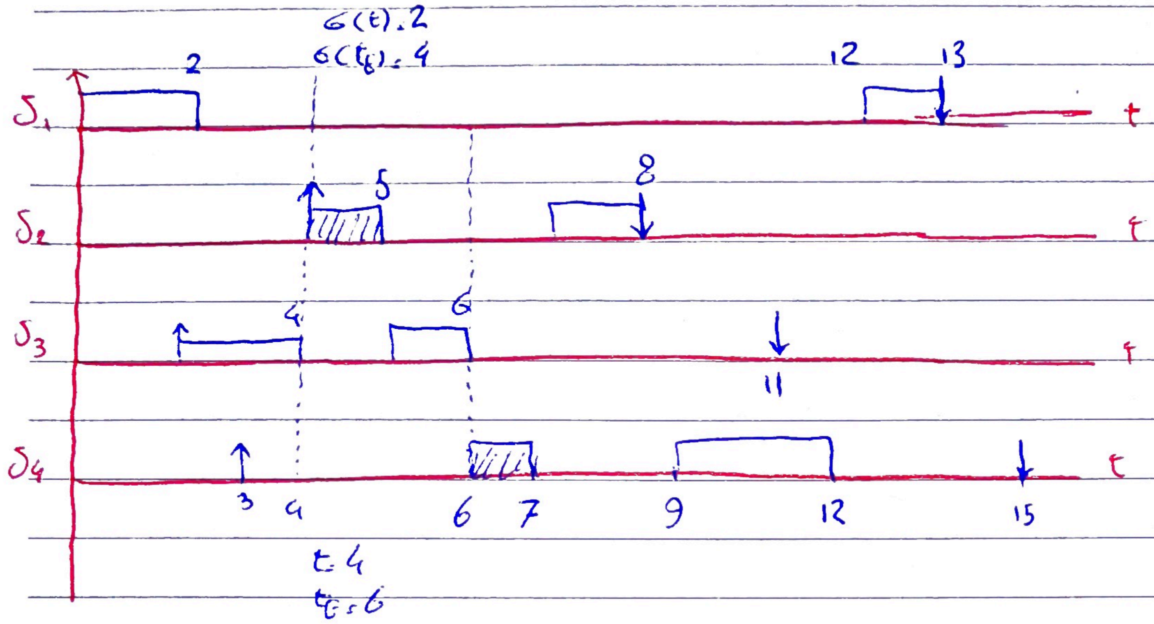
حال ۲ نمودار برای درک بهتر آورده‌ایم:

$\sigma(t)=4$
$\sigma(t_E)=2$

زمان‌بند σ در $t=4$:



زمان‌بند جدید:



الگوریتم transformation برای اثبات اپتیمال بودن EDF :

```
{
    for (t=0 to D-1){
        if(6(t) ≠ E(t)){
            6(t_E) = 6(t);
            6(t) = E(t);
        }
    }
}
```

برای هر بخش زمانی t این الگوریتم بررسی می‌کند آیا تسک $6(t)$ که در بازه t زمان‌بندی شده است، نزدیک‌ترین ددلاین را در بین تسک‌ها دارد یا خیر. اگر دارد که هیچ، ولی اگر این طور نبود پس جابجایی انجام می‌دهیم و بخش‌های زمانی های t و t_E جابجا می‌شوند و با توجه به توضیحات داده شده به راحتی می‌توان نشان داد که پس از هر جابجایی، maximum lateness نمی‌تواند افزایش یابد، پس EDF بهینه است.

اثبات کامل در آخر نوشته‌ها موجود است.

DATE / / SUBJECT:

با اعمال الگوریتم ذکر شده به schedule ای که در صفحات قبل رسم کردم، اولین جابجایی در t_1 رخ می‌دهد، در این لحظه پردازنده به j_2 اختصاص داده می‌شود اما وظیفه با نزدیک‌ترین ددلاین j_1 است که در t_1 تا t_2 زمانبندی شده است. بنابراین ۲ تکه‌های شور خورده با هم جابجا شده و زمانبندی جدید ایجاد می‌شود.

الگوریتم تمام برش‌های زمانی را بررسی می‌کند تا زمانی که idle شود و هر زمان که لازم شد جابجایی انجام می‌دهد.

همچنین برای اینکه نشان دهیم که جابجایی زمانبندی پذیری را حفظ می‌کند، توجه داریم که در هر لحظه هر برش زمانی در t می‌تواند یا جلو بیفتد و یا تا t_2 عقب بیفتد. در صورتی که جلو برود، feasibility شکل حفظ می‌شود اما اگر برش زمانی از شکل j_1 تا t_2 عقب بیفتد و t_1، feasible باشد، باید $t_2 \le d(j_1)$ باشد، مطلبی که j_1 زودترین ددلاین باشد. و چون $t_2 \le d(j_2)$ ، $t_2 \le d_2$ است داریم که $t_2 \le d(j_1)$ است که زمان بندی پذیری برش زمانی به تعویق افتاده را تضمین می‌کند.

* حال می‌خواهیم نشان دهیم هر یک از جابجایی‌هایی که در بخش داخلی الگوریتم رخ می‌دهد باعث نمی‌شود تا maximum lateness افزایش یابد.

فرض کنید در زمان t هستیم و $E(t)=a$ و $G(t)=b$ است. فرض کنید یک گام از الگوریتم را انجام داده و به زمانبندی [illegible] رسیده‌ایم. حال ۲ حالت خواهیم داشت:

1 فقط یک واحد از b و a باقی مانده باشد

$L_a = F_a - d_a$

$L_b = F_b - d_b$

$\rightarrow L'_a = F_b - d_a$

$L'_b = F_a - d_b \xrightarrow[d_b > d_a]{F_a > F_b} L'_b < L_a \, ، \, L'_a < L_a$

در نتیجه $\max(L'_a, L'_b) > \max(L_a, L_b)$.

۲) یک واحد از a و چند واحد از b برنده باشد، این حالت خود ۲ بخش است:

* $r_a < r_b$:

$$L_a = r_a - d_a \quad \wedge \quad L'_a = (t+1) - d_a \quad \Rightarrow \quad r'_a = t+1 < r_a < r_b$$
$$L_b = r_b - d_b \qquad\quad L'_b = (r_b) - d_b \quad \Rightarrow \quad L'_a < L_a \ ,\ L_b = L'_b$$

$\max(L'_a, L'_b) \le \max(L_a, L_b)$.

* $r_a > r_b$:

$$d_a < d_b \longrightarrow L'_b < L_a$$
$$t+1 < r_a \longrightarrow L'_a < L_a \quad \Rightarrow \quad \max(L'_a, L'_b) \le \max(L_a, L_b)$$

۳) اول واحد از b و چند واحد از a برنده باشد.

$$L_a = r_a - d_a$$
$$L_b = (t+1) - d_b$$
$$L'_a = r_a - d_a \quad \xrightarrow[d_b > d_a]{t+1 < r'_b < r_a} \quad L_a = L'_a$$
$$L'_b = r'_b - d_b \qquad\qquad\qquad L'_b < L_a \Rightarrow \max(L'_a, L'_b) \le \max(L_a, L_b)$$

۴) از هر دو، چند واحد برنده باشد که به ۲ حالت تقسیم می‌شود:

* $r_a > r_b$:

$$L_a = r_a - d_a$$
$$L_b = r_b - d_b$$
$$L'_a = r_a - d_a \qquad \xrightarrow[L'_b < L_a,\ L_a = L'_a]{r_a > r'_b} \qquad \max(L'_b, L'_a) < \max(L_a, L_b)$$
$$L'_b = r'_b - d_b$$

DATE / / SUBJECT:

* $F_b > F_a$:

$L_a = F_a - d_a$

$L_b = F_b - d_b \longrightarrow \max(L_a', L_b') \leq \max(L_a, L_b)$

$L_a' = F_a - d_a$

$L_b' = F_b - d_b$