Contents

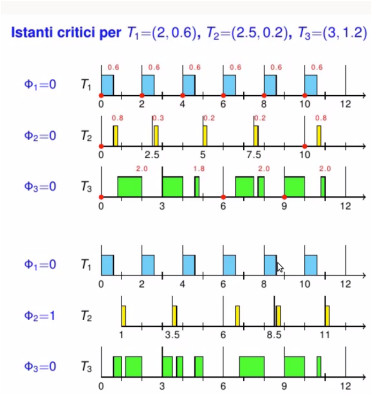
1 Schedulabilità di algoritmi a priorità fissa

Algoritmi a priorità dinamica, come EDF, sono ottimali (sotto determinate condizioni): se $\exists$ schedulazione fattibile $\Rightarrow$ anche EDF trova schedulazione.
Nessun algoritmo X a priorità fissa può avere un fatto di utilizzazione $U_X = 1$, deve per forza essere < 1.
Inoltre RM è ottimale (in senso assoluto, ovvero può raggiungere U = 1) per sistemi armonici con scadenze implicite.
In questa condizione RM è tanto buono quanto EDF.
DM è ottimale tra gli algoritmi a priorità fissa, ma non in senso assoluto: se $\exists$ algoritmo a priorità fissa che trova una schedulazione fattibile per un insieme di task, allora lo fa anche DM. Questo mi fa capire assegnare priorità fisse ai task,in modo arbitrario, non fa guadagnare nulla rispetto ad assegnarle con un parametro come la scadenza relativa. Algoritmo è altrettanto buono, se non più buono di algoritmi che fissano le scadenze in modo soggettivo, posso realizzare sistemi di task basati su parametri oggettivi e non soggettivi.
Corollario: RM è ottimale tra gli algoritmi a priorità fissa per sistemi ti task con scadenza proporzionale al periodo.
Mi pongo un problema generale: se ho sys di task generale ed un algoritmo di schedulazione a priorità fissa, come faccio a verificare il sistema, ovvero a certificare che l'algoritmo produrrà sempre una schedulazione valida?

1.1 Istanti critici

Istanti critici: suppongo che nel sys di task tutti i job abbiano un tempo di risposta piccolo, ovvero ogni job termina prima del rilascio del job successivo del task $\Rightarrow$ ogni job viene rilasciato in un periodo e si conclude entro quel periodo (job potrebbe non rispettare la scadenza, se questa è minore del periodo).
L'istante critico è il momento in cui il rilascio del job comporta il massimo tempo di risposta possibile per quel job.
Se almeno un job T_i non rispetta la scadenza relativa, l'istante critico è un momento in cui il rilascio di un job provoca il mancato rispetto della scadenza di quel job.
Io voglio verificare che tutti i job rispettino le scadenze, sottigliezza della definizione è irrilevante dal punto di vista critico.

Teorema: Se ho un sistema di task a priorità fissa e tempi di risposta piccoli, l'istante in cui uno dei job di T_i viene rilasciato contemporaneamente ai job di tutti i task con priorità maggiore di T_i è l'istante critico di T_i.
Teorema non da condizione necessaria e sufficiente, ma solo sufficiente: se capita tale condizione $\Rightarrow$ ho un istante critico, ma potrei averne altri.
esempio : T_1=(2, 0.6) T_2=(2.5,0.2), T_3=(3,1.2).
T_1 ha la priorità massima: tutti i multipli di 2 sono istanti critici.
T_2 ha istanti critici 0 e 10, che sono anche i momenti in cui rilascio job di T_1, non c'è nessun altro momento in cui c'è rilascio contemporaneo di job di T_1 e T_2.
T_3 avrà rilasci in 0, 3, 6, 9: in 0 ho istante critico, 6 e 9 sono critici ma il thm non li evidenzia.



Stesso esempio anche se i task non sono in fase: 6 è istante critico, è descritto dal teorema.
Quando c'è rilascio in fase, siccome priorità è fissa, la schedulazione prodotta risulta identica a qualsiasi schedulazione non in fase $\Rightarrow$ mi interessa ricondurmi a quando tutti i task sono in fase.

1.2 Schedulabilità per priorità fissa e tempi di risposta piccoli

Supponiamo che in un sistema ho task a priorità fissa e tempi di risposta piccoli. Ordino i task per priorità decrescente, suppongo siano in fase all'istante t_0.
Ho i task T_1.....T_i e mi chiedo il tempo necessario per eseguire tutti i job dei task T_1.....T_i, nell'intervallo $[t_0, t_0+t]$ ($t \leq p_i$:

$$w_i(t) = e_i + \sum_{k=0}^{i-1} \lceil \frac{t}{p_k} \rceil \cdot e_k.$$

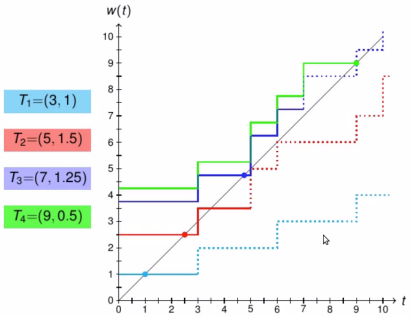
Somma si estende su tutti i task di priorità superiore di T_i, devo considerarli perché portano via tempo al job di T_i. Prendo k-esimo task: a t_0 tutti i task sono in fase, quindi rilascio sicuro un job, quando ne rilascio? Prendo il ceil di $\frac{t}{p_k}$,anche job rilasciato nel periodo dopo quello considerato mi ruba tempo; moltiplico tutto per e_k, il tempo che ci metto per completare i job.
Test di schedulabilità: dati job T_1.....T_i, in fase a t_0 con priorità decrescenti

con T_1.....T_{i-1} effettivamente schedulabili. Il task T_i può essere schedulato nell'intervallo di tempo [t_0, t_0+D] se $\exists$ t $\leq D_i$ tale che $w_i(t) \leq t$. IL mio scopo è sempre quello di verificare la schedulabilità del sistema, se ne trovo uno non schedulabile la mia analisi è finita, non ci faccio nulla col sistema di task.
Applicazione: ho T_1.....T_n con priorità decrescenti.
Considero un task alla volta: $\forall$ task T_i calcolo il valore della funzione di tempo necessario $w_i(t)$ per tutti i valori t $\leq D_i$ tali per cui t è un multiplo intero di p_k per k $\in$ {1,2....,i }.Funzione $w_i(t)$ sale a gradini, devo considerare valori per cui tale funzione cambia valori.
Se per almeno uno dei valori t vale che $w_i(t) \leq t$ allora T_i è effettivamente schedulabile. Altrimenti il test fallisce, ovvero n job di T_i potrebbe mancare la scadenza, ovvero la manca sicuro se c'è un rilascio di tutti i job in fase dei task di priorità superiore e tutti quei task hanno un tempo di esecuzione pari al loro worst-case.
Possono esserci casi fortuiti, quindi in ipotesi rilassate il test non conferma schedulabilità ma scheduler riesce, però il risultato non è rilevante.
Tanto vale fermarsi e riprogettare il sistema.
esempio: T_1=(3,1), T_2=(5,1.5), T_3=(7, 1.25), T_4=(9,0,5) e considero le funzioni di tempo necessario:

Esempio: T_1=(3, 1), T_2=(5, 1.5), T_3=(7, 1.25), T_4=(9, 0.5)

t	3	5	6	7	9
$w_1(t)$	1.0				
$w_2(t)$	2.5	3.5			
$w_3(t)$	3.75	4.75	6.25	7.25	
$w_4(t)$	4.25	5.25	6.75	7.75	9.0

Grafico per l'esempio precedente, ho la bisettrice del 1° quadrante, dire che $w_i(t)$ è $\leq$ t vuol dire che $w_i(t)$ sta sotto la bisettrice. La funzione è a scalini, non ha senso calcolarla, la applico nel periodo tra 0 e la fine del periodo.In T_2 la funzione sale sopra la bisettrice, ma non è importante: devo verificare che sia sotto in un certo momento, se fosse sempre sopra non sarebbe schedulabile.
Ogni volta che c'è rilascio di un task a priorità superiore $\Rightarrow$ ho gradino nella funzione di tempo necessario.



1.3 Massimo tempo di risposta

Massimo tempo di risposta W_i di T_i è il più piccolo valore prima della scadenza relativa t.c : t=w_i(t). Se l'equazione non ha soluzioni $\leq$ a D_i, allora qualche job di T_i mancherà la scadenza relativa.
Uso un algoritmo:

- $t^(1) = e_1$ in prima approssimazione
- Sostituisco nella funzione ed ottengo un nuovo valore $t^(k+1) = w_i(t^(k))$
- continuo ad iterare finché:
  - $t^(k+1) = t^(k)$ e $t^(k) \leq D_i \Rightarrow W_i = t^(k)$
  - $t^(k) \geq D_i$ e allora sono fuori scadenza

Ma dato che caso peggiore sono task in fase e dato che ho tutti i parametri sono noti, non sarebbe più facile provare a simulare la schedulazione? Sì, ma ci sono dei fattori che non ho considerato e che mi impediscono di simulare, esempio:

- Non è possibile determinare facilmente il worst case
- Il worst case cambia da task a task
- È difficile integrare nella simulazione altri fattori che possono essere considerati estendendo il test di schedulabilità.

In ogni caso, sia simulare il test che il test di schedulabilità stesso hanno la stessa complessità.

1.3.1 Task periodici con tempi di risposta arbitrari

Considero ora task con tempi di risposta arbitrari, che implica che:

- Un job non deve necessariamente prima che il job successivo dello stesso task sia eseguito
- è possibile che $D_i \geq$ di p_i
- Ci possono essere nello stesso istante più job di uno stesso task in attesa di essere eseguiti.
- Un job rilasciato contemporaneamente a tutti i job dei task con priorità maggiore non ha necessariamente il massimo t. di risposta possibile.

Assumo sempre che i job di uno stesso task hanno vincoli di precedenza impliciti fra di loro, ovvero sempre eseguiti FIFO.
Analizzo task per task: considero T_i (i precedenti sono schedulabili). Ho insieme task τ_i=T_1....T_i con priorità decrescente.
Definisco un intervallo totalmente occupato di un livello π_i un intervallo (t_0, t_1] tale che:

- all'istante t_0 tutti i job di τ_i rilasciatiti prima di t_0 sono stati completati
- All'istante t_0 un job di τ_i viene rilasciato.
- L'istante t_1 è il primo istante in cui tutti i job di τ_i rilasciati a partire da t_0 sono stati completati

È possibile che in un intervallo totalmente occupato il processore sia idle o esegua task non di τ_i? No: se fosse idle, l'intervallo terminerebbe prima, non può neanche eseguire task di priorità inferiore, quindi non può eseguire task al di fuori di τ_i
esempio: T_1, T_2, T_3.
Intervalli di T_3 non sono lunghi uguale, questo perché i rilasci di T_3 non sono in concomitanza con T_1 e T_2, posso dire che l'intervallo a lunghezza massimo quando i rilasci di tutti i task sono in fase.
Test di schedulabilità generale per tempi di risposta arbitrari è ancora basato sul caso peggiore, la differenza rispetto al test per tempi piccoli è che il primo job rilasciato contemporaneamente agli altri potrebbe non avereil massimo tempo di risposta.
Idea : $\forall$ T_i analizzo tutti i suoi job eseguiti nel primo intervallo totalmente occupato di livello π_i.
Come determino l'intervallo totalmente occupato:

- Inizio determinato dal rilascio dei primi job (in fase) dei task τ_i={T_1, ..., T_i}
- Lunghezza massima calcolata risolvendo iterativamente $t = \sum_{k=1}^{i} \lceil \frac{t}{p_k} \rceil \cdot e_k$. Molto simile alla funzione di tempo necessario, dico che aumento t fino a che non trovo il valore dato dalla sommatoria, ovvero il primo t per cui il lavoro necessario per compiere tutti i task permette di eseguire tutti i task rilasciati nell'intervallo [t_0, t_0+t]

Quindi si procede nel seguente modo:

- Considero i task {T_1, ..., T_i} con priorità $\pi_1 < \pi_2 .... < \pi_i$, considero un task T_i alla volta cominciando da quello con la massima priorità, ovvero T_1
- Il caso peggiore per la schedulabilità di T_i: assumere che i task τ_i = {T_1, ..., T_i} sono in fase.
- Se il primo job di tutti i task in Tau_i termina entro il primo periodo del task $\Rightarrow$ decidere se T_i è schedulabile si effettua controllando se $J_{i,1}$ termina entro la scadenza tramite la funzione di tempo richiesto $w_{i,1} := w_i(t)$
- Altrimenti almeno un primo job di Tau_i termina dopo il periodo del task, calcola la lunghezza t^L dell'intervallo totalmente occupato di livello π_i che inizia da t = 0.

- Calcolo i tempi di risposta massimi di tutti i job di T_i dentro l'intervallo totalmente occupato che sono $\lceil \frac{t^L}{p_i} \rceil$; il primo l'ho già calcolato.
- Decido se questi job sono schedulabili dentro l'intervallo totalmente occupato. Uso un lemma:
  Il tempo di risposta massimo $W_{i,j}$ del j-esimo job di T_i, in un intervallo totalmente occupato di livello π_i in fase è uguale al minimo t che soddisfa l'equazione $t = w_{i,j}(t+(j-1)\cdot p_i) - (j-1)\cdot p_i$, con $w_{i,j}(t) = j\cdot e_i + \sum_{k=1}^{i-1} \lceil \frac{t}{p_k} \rceil \cdot e_k$.
  Aggiungo un j che moltiplica e_i, devo verificare l'equazione nei punti multipli.

esercizio: $T_1 = (\phi_1,2,1,1)$, $T_2 = (\phi_2,3,1.25,4)$, $T_3 = (\phi_3,5,0.25,7)$
Parto verificando T_1:
$w_1(t) = w_{1,1}(t) = e_1 = 1 = D_1$. Quindi è sicuramente schedulabile .
T_2:
$w_{2,1}(2) = e_1 + e_2 = 2.25 > 2$, quindi non va bene. Vado avanti:
$w_{2,1}(3) = 2\cdot e_1 + e_2 = 3.25 > 3$. Non va ancora bene, proseguo:
$w_{2,1}(4) = 2\cdot e_1 + e_2 = 3.25 \leq 4 \leq D_2$ quindi T_2 è schedulabile, ma ha completato oltre il periodo $\Rightarrow$ non posso più considerare tempi piccoli, devo considerare gli intervalli totalmente occupati, uso l'equazione iterativa:
$t^(1) = e_1 + e_2 = 2.25$, sostituisco nella sommatoria,ed ottengo $t^(2) = 2\cdot e_1 + e_2 = 3.25$, $t^(3) = 2\cdot e_1 + 2\cdot e_2 = 4.5$, $t^(4) = 3\cdot e_1 + 2\cdot e_2 = 5.5$, $t^(5) = 3\cdot e_1 + 3\cdot e_2 = 5.5 \Rightarrow t^(4) = t^L$, ovvero intervallo totalmente occupato di livello 2 è 5.5.
Ora calcolo quanti job di T_2 ci sono in $(0, 5.5] = \lceil \frac{t^L}{p_2} \rceil = 2$.
Veridico il secondo job di T_2:
$w_{2,2}(3) = 2\cdot e_1 + 2\cdot e_2 = 4.5 > 3$, no
$w_{2,2}(4) = 2\cdot e_1 + 2\cdot e_2 = 4.5 > 4$, ancora no.
$w_{2,2}(3) = 3\cdot e_1 + 2\cdot e_2 = 5.5 \leq 6 \leq p_2+D_2=7$, quindi accetto il task.
Ora devo capire se posso accettare T_3, e considerare l'intervallo totalmente occupato di lvl 3:
$t^(1) = e_1 + e_2 + e_3 = 2.5$
$t^(2) = 2\cdot e_1 + e_2 + e_3 = 3.5$
$t^(3) = 2\cdot e_1 + 2\cdot e_2 + e_3 = 4.75$
$t^(4) = 3\cdot e_1 + 2\cdot e_2 + e_3 = 5.75$
$t^(5) = 3\cdot e_1 + 2\cdot e_2 + 2\cdot e_3 = 6$
$t^(6) = 3\cdot e_1 + 2\cdot e_2 + 2\cdot e_3 = 6 = t^L$
job di T_3 nell'intervallo (0,6]: $\lceil \frac{t^L}{p_3} \rceil = 2$. Considero i singoli job:
$w_{3,1}(2) = e_1 + e_2 + e_3 = 2.5 > 2$, no.
$w_{3,1}(3) = 2\cdot e_1 + e_2 + e_3 = 3.5 > 3$, no.
$w_{3,1}(4) = 2\cdot e_1 + 2\cdot e_2 + e_3 = 4.75 > 4$, no.
$w_{3,1}(5) = 3\cdot e_1 + 2\cdot e_2 + e_3 = 5.75 > 5$, no.
$w_{3,1}(6) = 3\cdot e_1 + 2\cdot e_2 + e_3 = 5.75 \leq 6 \leq D_3 = 7$. Posso accettare il job

$w_{3,2}(5) = 3\cdot e_1 + 2\cdot e_2 + 2\cdot e_3 = 6 > 5$, no.

$w_{3,2}(6) = 3 \cdot e_1 + 2 \cdot e_2 + 2 \cdot e_3 = 6 \leq 6 \leq p_3 + D_3 = 12$ Accetto il job, e quindi il task.
Tutti i task sono schedulabili a prescindere dai loro task.

1.4 Condizioni di schedulabilità

Il test di schedulabilità generale determina se insieme di task è schedulabile o no, considerando worst case che è task in fase.
Ho dei limiti:

- Devo conoscere tutti i periodi, le scadenze ed i tempi d'esecuzione. Per validazione è necessario, ma no per implementazione di scheduler a priorità fissa. Se voglio aggiungere un task dovrei conoscere parametri che in fase di progettazione del sw non servono.
- Il risultato ottenuto non è valido se il task varia periodo, scadenza o tempo di esecuzione.
- È computazionalmente costoso, poco adatto per scheduling on-line.

Cerco di trovare delle condizioni di schedulabilità, confronto il test con la condizione, che è molto più semplice da calcolare e che può essere applicata anche se alcuni parametri non sono noti (esempio: condizione di EDF).
Mi chiedo se $\exists$ condizione di schedulabilità per algoritmi a priorità fissa:
Condizione di Liu-Layland: sistema τ di n task indipendenti ed interrompibili con scadenze relative uguali ai rispettivi periodi può essere effettivamente schedulato su un processore in accordo con RM se il suo fattore di utilizzazione U_τ è $\leq$ a $U_{RM}(n) = n \cdot (2^{\frac{1}{n}} - 1)$
Questo è il fattore di utilizzazione di RM, se considero: $\lim_{n \to \inf} U_{RM}(n) = \ln 2$, ovvero RM in generale garantisce di rispettare le scadenze pur di non caricare il processore per più del 69.3.
Ho un criterio per adottare RM negli scheduler real-time.
esempio:
$T_1 = (1,0.25)$, $T_2 = (1.25,0.1)$, $T_3 = (1.5,0.3)$, $T_4 = (1.75,0.07)$, $T_5 = (2,0.1)$.
$U_\tau = 0.62 \leq 0.743 = U_{RM}(5) \Rightarrow$ è schedulabile con RM.
IL sistema $T_1 = (3,1)$, $T_2 = (5,1.5)$, $T_3 = (7,1.25)$, $T_4 = (9,0.5)$ ha fattore di utilizzazione $U_\tau = 0.867 > 0.757 = U_{RM}(4)$, forse non schedulabile.
È condizione sufficiente, difatti l'esempio 2 era quello precedente che è schedulabile se applico la funzione di tempo necessario.
L'alternativa a questo risultato è il test iperbolico: Un sistema τ di n task indipendenti ed interrompibili con scadenze relative uguali ai rispettivi periodi può essere effettivamente schedulato su un processore RM se $\prod_{k=1}^{n} (1 + \frac{e_k}{p_k}) \leq 2$.
SI applica anche questo conoscendo solo fattore di utilizzazione dei task.
Correlazione con condizione di Liu-Layland: se gli n task hanno tutti lo stesso rapporto $\frac{e_k}{p_k}$ vuol dire che ciascun di questi usa una porzione uguale del processore.

Si può dimostrare che se questo è vero allora, assumendo $u_k = \frac{U_\tau}{n}$:
$\prod_{k=1}^{n}(1+\frac{e_k}{p_k}) \leq 2 \Leftrightarrow U_\tau$ *leq* n·($2^{\frac{1}{n}}$-1).
Se questo non è vero, esistono casi in cui il test iperbolico è soddisfatto, ma la condizione di Liu-Layland no; non esiste invece mai il viceversa.

1.5 Test per sottoinsiemi di task armonici

So che ,in generale RM è schedulabile se è soddisfatta condizione di Liu-Layland, ma so anche che su task armonici è ottimale. Suddivido insiemi di task in sottoinsiemi di task armonici fra loro.
Condizione di Kuo-Mok: se sistema τ di task periodici, indipendenti ed interrompibili con $p_i = D_i$ può essere partizionato in n_h sottoinsiemi disgiunti Z_1,....,Zn_h, ciascuno dei quali contiene task semplicemente periodici, allora il sistema è schedulabile con RM se:
$\sum_{k=1}^{n_h} U_{Z_k}(n_h)$ oppure se $\prod_{k=1}^{n_h}(1+U_{Z_k}) \leq 2$.
Se un sistema ha poche applicazioni molto complesse, è possibile migliorare la schedulabilità rendendo i task di ciascuna applicazione semplicemente periodici.
Esempio: 9 task con periodi 4,7 ,7 , 14, 16, 28, 32, 56, 64, fattore di utilizzazione di Liu-Layland è $U_{RM} = 0.720$
Considero i multipli di 2 e 7 e partizionando in due sottoinsiemi ottengo $U_{Z_1} + U_{Z_2} \leq U_{RM}(2) = 0.828$.

Il fattore di RM è in generale U_{RM}(n), ma posso farlo diventare pari ad 1 per task semplicemente periodici.
Miglioro U_{RM}(n) considerando quanto i periodi dei task sono vicini ad essere armonici:
$X_i = \log_2 p_i$ - $\lfloor \log_2 p_i \rfloor$ e $\zeta = \max_{1\leq i\leq n} X_i$ - $\min_{1\leq i\leq n} X_i$
Considero il valore frazionario del $\log_2$ e prendo tutti i task, di cui faccio differenza tra max e min di questi scarti decimali.
Teorema: nelle ipotesi della condizione di Liu-Layland, il fattore di utilizzazione di RM dipende dal numero di task n e da ζ è: $U_{RM}(n, \zeta) =$

- (n-1)·($2^{\frac{\zeta}{(n-1)}}$-1) + $2^{(1-\zeta)}$-1 se $\zeta < 1 - \frac{1}{n}$
- U_{RM}(n)

Quando si verifica il caso $\zeta = 0$? Quando $p_i = K\cdot 2^{x_i}$; non è vero il contrario
Variante: schedulabilità per scadenze arbitrarie. Se per qualche task la scadenza è più grande del periodo il limite è valido? Sì, però la formula è "pessimista": forse è possibile trovare valori di soglia superiori a U_{RM}.
Se invece per qualche task il periodo è più grande della scadenza non posso applicare Liu-Layland.
Teorema:
Un sistema τ di n task indipendenti, interrompibili e con scadenze $D_i = \delta p_i$ è schedulabile con RM se U_τ è $\leq$ a: $U_{RM}(n, \delta) =$

- δ(n-1)$\cdot(\frac{\delta+1}{\delta}^{\frac{1}{(n-1)}}$ - 1) per δ = 2,3,.....
- n($2\delta^{\frac{1}{n}}$-1) + 1 - δ per $0.5 \leq \delta \leq 1$
- δ per $0 \leq \delta \leq 0.5$